Designing and Evaluating an Educational Board Game

UNIVERSITY
OF TWENTE.

Steven de Heus

March 4, 2020

Abstract

We design and create an educational board game based on the mathematical concept of parity games. We explore the research areas of computational, mathematical and systems thinking and discuss how the game helps the players train these thinking styles. We test the game in a school pilot with 44 students from age 8 to 12 from three different elementary schools. Finally, we create a simulation of the game and implement a computer player. Using an evolutionary algorithm we find an optimal configuration for the computer player. We analyze the data gathered from the school pilot and the play patterns made by the computer player. The main findings are that the game is suitable for the chosen age group in terms of complexity, challenge and entertainment. Additionally, we recommend a number of ways the game and other future educational board games could be improved.

Contents

Chapter 1

Introduction

Parity games are a type of mathematical problem. Two players move a shared token along a directional graph with numbered nodes. Each node is associated with a player, that player makes the next move when the token is in that node. You win the game by forcing the token into an infinite loop where the highest number on the nodes has your parity (odd or even).

We set out to make an educational board game based on the concept of parity games. First we explore what educational benefits such a game might serve outside of parity games and graph theory. To do this we turn to the research areas of computational, mathematical and systems thinking. After this preliminary study we establish a set of requirements and design and create the game through an iterative process. We take the game to three elementary schools to see how suitable it is for students from age 9 to 12 and to observe how they play. Finally, we create a model of the game and develop a computer player to play the game in simulations. Based on the data of the school pilot and the simulations we make a number of recommendations for the board game and educational games in general, specifically for the research area of computational thinking.

1.1 Preliminaries

In the preliminary study we explore related work. The main focus is computational thinking. Jeannette Wing [20] defines computational thinking as the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out. We also discuss Piaget's theory of cognitive development and investigate how mathematical and systems thinking relate to computational thinking. Finally we take a look at other studies about educational games.

1.2 Game Design

Like tic tac toe, parity games are a perfect information game. That means both players know everything and can theoretically determine who is going to win before the first player even makes a move. This is not a problem when the players have a lot of options for their moves in games like chess, but a small parity game would not be much more interesting than playing tic tac toe. Even a larger parity game would eventually be solved completely and not be interesting for players who have played the game several times before. This is why we added an element of hidden information and randomness by giving the players a hand of cards. Players can not see the cards other players

have in hand and can play cards to change the numbers on the nodes and set a strategy for how the token will move.

Instead of using parity (whether a number is odd or even), we use color. A node can have a number of colored coins on it. The color indicates who the coins belong to and a node can only have coins of one color. We made the decision to use colors instead of parity so more than two people can play the game and because it is more intuitive.

Once all the cards are played, the token moves in a deterministic way across the graph and the score is counted. During the final turns of the game the players still have perfect information if they counted the cards that were played, which is doable with the small number of cards there are.

We create an initial design with this idea and make a paper prototype. We test this prototype and improve it during three iterations. During this process we test several mechanics, rules, board layouts and cards. We select the best design and create five copies of the game to be used in the school pilot.

1.3 School Pilot

We visit three elementary schools and test the game with 44 students. Groups of 5 to 8 students spend around an hour participating in the pilot. During this hour they learn how the game works, play a few rounds and finally they take a survey. The survey asks the students what they thought of the game. It aims to measure how much they enjoyed it, how complex the game was and how challenging or exciting it was.

We analyze the data from the survey and test a list of hypotheses. Aside from the survey, we observe how the students play: what sort of strategies they use and what kind of mistakes they make. Based on these results we draw conclusions about the game and make recommendations for future work.

1.4 Simulation and Computer Player

The simulation serves three purposes: to find ways to improve the game, to demonstrate how to express the problem (and solution) of finding a good strategy in a way that a computer can execute it (in other words, how to apply computational thinking to the game) and to lay a foundation for future work of checking more complex properties in games like having a snowball effect (see section 7.3 on page 58).

We create a Python program that tracks all the details of the state of the game. We implement a computer player (for 2 player games) that will determine a move for any given game state. The computer player distinguishes three different stages of the game:

1. There are still cards in the deck. At the end of every turn the active player draws a card from the deck if possible. When there are still cards in the deck the computer player does not know which cards the other player has in hand. It will use several heuristics and parameters to come up with a reasonable move. The parameters of this stage play an important role in the evolutionary algorithm later on.
2. The deck is empty but the players still have a lot of cards in hand. During this stage the computer player knows what cards the other player has in hand, but there are too many remaining turns to calculate the best move. Instead, it will calculate every possibility for up to 3 moves into the future and picks the best one it can find.

3. The deck is empty and the players each have 2 or less cards in hand. During this stage the computer player can calculate the best move for both players for last few turns of the game.

We use an evolutionary algorithm to find a local optimum for the parameters of the first stage of the computer player. Finally we draw conclusions based on the local optimum that was found.

1.5 Research Goal and Questions

The goal of this project is to design and create a board game and to evaluate its potential to be used as an educational tool. The game will be evaluated in two ways: a pilot with elementary school students and a simulation with a computer player. We gather and analyze data from the pilot as well as the simulation.

1.5.1 Research Questions

We aim to answer the following research questions:

1. How suitable is the game for students from age 9 to 12?
2. How do factors such as age, gender, group size, game outcome, board game experience and competitiveness impact the experience of the students playing the game? The 'experience' is described in terms of how fun, challenging and complex the game is perceived.
3. How does the game relate to computational thinking?
4. What strategies do the students in this study use when playing the game?
5. How can we use the data from the pilot and the simulations to improve this game?

1.5.2 Motivation

For the field of computational thinking in education an important step forward would be to develop a standardized method of measuring computational thinking skills in students. This could be done with a test where students solve various age appropriate mathematical (or other types of) problems. This would allow smaller studies like this one to directly measure the effectiveness of a potential teaching method (such as board games). Unfortunately, creating such a method is outside of the scope of this project. Instead, we will focus on things we *can* measure and are useful for future research in this area. For example, if a game is considered to be used in education, it should have a good balance of complexity, challenge and fun to maximize its potential. These aspects of the play experience can be measured with a survey. Another thing we can measure and analyze is the strategies which students use by observing them play the game.

According to Piaget's theory of cognitive development (as discussed in the section 2.1.2) children start to develop abstract reasoning around the age of 11. Ideally the game is suitable for children who are about to start developing their abstract reasoning skills as well as older children and adults. To test this, we have chosen to do the school pilot with students from age 9 to 12.

By collecting and analysing all this data we aim to draw useful conclusions for future research in this area. For example, we may find that this game is too complex for the younger students in this study because of a certain aspect of the game. That would allow future studies to avoid using similar game aspects if they want to target students of the same age. Or we could find that students mostly enjoyed one on one games, or three or more player games. Maybe the students will not enjoy the competitive aspect at all, which means future research should be focused more on cooperative games where the players work together towards a common goal.

The simulation part is included for a number of reasons. By building a model and running various simulations we gain a better understanding of the game and the flow and patterns of the gameplay. We can use the model to prove simple properties of the game, such as that it always terminates. The simulation and the computer player also serve as a demonstration on how to use abstract reasoning (an important computational thinking skill) to create strategies for the game.

1.6 Contributions

The main contribution of this study is a board game that is suitable for the chosen age group (age 9 to 12). The students indicate they enjoy the game a lot. We observe that they understand the basic principle of the game and the concept of cycles. However there are still several tactics they did not deploy, indicating there is still room for growth by playing the game more. We also demonstrate how we applied computational thinking to create a simulation and a computer player. Finally we use the data gathered from the school pilot and the simulations to make recommendations for improving this game and educational games in general.

Chapter 2

Preliminaries

2.1 Related Work

In this section we discuss current status of the research area of computational thinking, board games in education and similar styles of thinking. We first examine computational thinking and then examine mathematical thinking and systems thinking. We show how these thinking styles are present in the proposed game in section 3.8.

2.1.1 Computational Thinking

Although the concept of computational thinking dates back many years, it was brought to attention again by Jeannette M. Wing in 2006 [19] as ”It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.” There is no universally agreed upon definition of what exactly computational thinking is, but there are several noteworthy definitions:

- In the same paper from 2006 [19], Wing offers a whole host of examples and definitions such as:
  - ”Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation.”
  - ”Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system. It is separation of concerns. It is choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. It is using invariants to describe a system’s behavior succinctly and declaratively. It is having the confidence we can safely use, modify, and influence a large complex system without understanding its every detail. It is modularizing something in anticipation of multiple users or prefetching and caching in anticipation of future use.”
- In the summary of a presentation in 2012 [9] Wing, Cuny and Snyder state that ”Computational thinking is the thought processes involved in formulating problems and expressing its solution as transformations to information that an agent can effectively carry out.” In this context, an agent could be a computer, another type of machine or simply a person following a set of instructions.

- In 2015, Atmatzidou and Demetriadis [14] carried out a study on the effect of doing seminars with a robotics educational kit on the computational thinking skills of students aged 15 and 18. They use a model of computational thinking which breaks it down into five core skills:
  - Abstraction: separating important from redundant information to find the relevant patterns and important ideas.
  - Generalisation: expanding an existing solution to cover more cases.
  - Algorithm: identifying effective algorithms and finding the most efficient algorithm to solve a problem.
  - Modularity: developing a set of often used actions/steps/commands that can be reused to solve different problems.
  - Decomposition: breaking down a problem into smaller, simpler parts that are easier to manage.
- In a paper from 2016, David Weintrop et al. [16] divide computational thinking into four categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices.
  - Data practises are about how data is collected, created, analyzed, manipulated and shared.
  - Computational problem solving practises involve preparing problems to be solved by computers, computer programming, choosing the right computational tools, troubleshooting problems, developing modular solutions and creating computational abstractions.
  - Modeling and simulation practices involve constructing computational models, using computational models to understand a concept, using computational models to find and test solutions and assessing computational models.
  - Systems thinking practises involve investigating a complex system as a whole, understanding the relationships within a system, thinking in levels and communicating information about a system.
- In 2017, Shute et al. published a thorough literature study on many different computational thinking definitions and models and compared computational thinking to other thinking styles such as mathematical thinking [13]. They created their own definition of computational thinking: ”the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts”. They have six facets of computational thinking: decomposition, abstraction, algorithms, debugging, iteration and generalisation. Most of these are the same as previously mentioned processes or skills. Debugging is to detect, identify and solve a problem. Iteration is to repeat design processes until the desired result is achieved.

There are several different definitions of computational thinking. The citations above are listed chronologically to show how the definitions used by different (or sometimes the same) authors changed over time. The definitions started out quite vague and broad at the start but eventually, most authors mostly have the same definition. The most common aspects of computational thinking from these definitions are abstraction, problem decomposition and algorithms. As stated in section 2.1.5, we will use the definition from Wing’s presentation in 2012 [9].

Since computational thinking re-emerged in 2006, there has been a lot of interest in integrating it into the curriculum of k12 and sometimes also college or university level education. A number of noteworthy projects and studies on this subject are listed below:

- The TangibleK program aims to teach simple robotics and programming concepts to young children. In 2010 it had already been piloted in prekindergarten to second grade [5]. The program offers the children a simple robotics kit that allows them to build robots and vehicles by snapping together parts. These are then programmed with simple instructions such as "(briefly) drive forward a number of times" or "turn on the light if the light sensor says it is dark". This teaches the children computational thinking concepts like modularity and algorithms. In 2013 the TangibleK program was tested in three kindergarten classrooms [4]. This study showed that the children were both able to and interested in learning about the aspects of robotics and programming.
- A paper by Yadav et al. in 2011 [21] describes the implementation and evaluation of a computational thinking module for education majors. It teaches future teachers what computational thinking is and how it can be taught without using computers. The results of this study suggest that the module makes education students' attitude towards computer science more favorable and it makes them more likely to integrate computing principles in future teaching.

2.1.2 Piaget's Theory on Cognitive Development

Abstract reasoning is an important aspect of most definitions of computational thinking. According to the widely used theory on cognitive development by Piaget [8] children and young adults undergo 4 stages of cognitive development:

1. Sensorimotor stage, infancy. During this stage the child mostly develops their motor skills through physical interations.
2. Pre-operational stage, toddler and early childhood. The child learns how to use a language and develops their memory and imagination.
3. Concrete operational stage, elementary and early adolescence. In this stage the child learns to use symbols and logic related to concrete objects (such as length, mass and volume).
4. Formal operational stage, adolescence and adulthood. During this stage the child or adult develops abstract reasoning ("intelligence is demonstrated through the logical use of symbols related to abstract concepts").

The overlap between computational thinking and Piaget's theory on cognitive development is the abstract reasoning of the formal operational stage. It is generally agreed that the formal operational stage starts between age 10 and 16 [2, 7, 10].

2.1.3 Systems Thinking & Mathematical Thinking

The literature study on computational thinking by Shute [13] also briefly discusses mathematical thinking and systems thinking (among other styles of thinking) and its relation to computational thinking. Mathematical thinking is the application of math skills to solve math problems. These math skills range from basic skills such as counting and arithmetic to more advanced skills like calculus and set theory. The overlap between Mathematical and computational thinking skills is problem solving, modelling, analyzing and interpreting data and statistics and probability.

The main idea behind systems thinking is to consider all the elements and relationships that exist in a system. Many problems can be analyzed in this manner to help with problem decomposition,

as they can be quite complex and dynamic. Systems thinking is heavily involved when modelling a system of any kind and using that model to run simulations. In particular, the systems thinking skills are (a) define the boundaries of a problem/system, (b) model/simulate how the system works conceptually, (c) represent and test the system model using computational tools, and (d) make decisions based on the model [12, 13].

Both of these thinking styles overlap with computational thinking and play a role in this project: the mathematical thinking skills counting, arithmetic and probability are used while playing the game, systems thinking is used because the players are constantly modelling the board as a system while playing the game.

Mathematical Thinking Activities in the Dutch High School Curriculum

In 2004 the Dutch Committee Future Mathematics Education (cTWO, based on the Dutch name 'Commissie Toekomst Wiskunde Onderwijs') was brought to life. Its goal was to design and test an improved mathematics curriculum for Dutch high schools. In 2015 the committee produced their final report [6]. The report stated that one of the six focus points for this reform was to include mathematical thinking activities ('wiskundige denkactiviteiten' in Dutch) as a common thread in all mathematics subjects. The report identifies the following mathematical thinking activities:

- Modelling and algebraization: modelling is the act of creating a fitting mathematical structure for a situation. Algebraization is to create formulas or equations to describe an aspect of a situation.
- Structuring and ranking: these abilities contribute to abstraction and analytical thinking.
- Manipulating formulas: the ability to change and solve formulas or equations.
- Abstraction: no definition given but likely similar to the abstraction aspect of computational thinking: separating important from redundant information to find the relevant patterns and ideas.
- Logical reasoning and proving: piecing together small parts of information to prove something.

The report also mentions that there has been too much focus on making students reproduce a short solution for a small mathematical problem instead of posing a larger problem that needs to be decomposed into smaller problems by the student, so problem decomposition is also an important skill for this report. In a way, this type of mathematical thinking is different from the mathematical thinking in section 2.1.3 and instead resembles a mix of computational thinking and mathematical thinking.

2.1.4 Board Games as Teaching Method

Board games (or games in general) are worth considering as a teaching method for a more computational thinking oriented curriculum. They are more engaging and fun than a conventional lecture. They challenge the players to come up with strategies to decompose and solve problems as efficiently as possible. When players repeatedly play the same game, they will start to recognize and respond to common play patterns. Games often present the players with a story and a complex system, so abstraction is required to understand the structure of the game. There have been many studies about the use of games for education in the context of computational thinking (or for more general purposes such as developing social skills), several noteworthy ones are listed below:

- A literature study by Hromek and Roffey from 2009 [11] argues that games are a good way of teaching social and emotional skills that are needed to work together in a respectful way

when negative emotions can be involved.

- Crabs & Turtles: a series of computational adventures [15] is a set of three board games that aims to teach coding concepts and computational thinking skills. This study found positive gaming experiences with these games when piloted by children aged 8 and 9. These games have a relatively explicit teaching style. For example, in one of the games the players move objects by creating sequences and loops of commands.
- In the game RaBit EscApe [1] the players create a path using magnetic building blocks (called bits) to help a rabbit escape from the apes. Although this game is not as explicit in teaching computational thinking skills as Crabs & Turtles, it was created for this purpose.
- In 2011, Matthew Berland and Victor Lee explored collaborative games, such as Pandemic, as a tool to foster computational thinking [3]. In this study, the communication between players of a collaberative game was categorized into several categories and plotted to find out how much computational thinking was going on. This showed that the players did in fact use computational thinking during gameplay and also which types of computational thinking they used.
- In South Africa, a Science, Education and Technology skill-building kiosk with computer games is used to target disadvantaged communities and increase their computational thinking skills [17]. This kiosk lets children play computer games aimed at improving computational thinking. In one of the games, players give a robot a sequence of commands to move it across a board. Another game lets the children play around with state machines.

All of the above games have a common goal: to teach computational thinking. Although some have studied the effectiveness and suitability of these games for the targeted group of students, most have not. Future work in this area should focus more on how these games are actually experienced, how suitable they are for students and what makes a game a good teaching tool or not.

2.1.5 Preliminary Study Conclusions

There has been a lot of interest in computational thinking since Wing wrote about it in 2006 [19]. By many it is considered a valuable set of skills that can help anyone, not just computer scientists, to solve problems. Researchers are making an effort to find methods to improve computational thinking skills in students and in the Netherlands the high school mathematics curriculum is being redefined to include several skills which are also important to computational thinking. Games and board games play an important role in these studies. They are engaging learning tools that often require the students to decompose problems and use abstract thinking to recognize gameplay patterns and find winning strategies.

The definition of computational thinking we use will be the one from Wing's presentation in 2012 [9]: "Computational Thinking is the thought processes involved in formulating a problem and expressing its solution in a way that a computer—human or machine—can effectively carry out."

This definition is selected as it was given by Jeanette Wing who arguably popularized computational thinking. We think it captures the essence of computational thinking, which is that it is about the skills used by computer scientists to solve problems with computers. It is exactly those skills that are useful for everyone, not just computer scientists, so we should make an effort to teach them to all students.

Chapter 3

Game Design

3.1 Parity Games

Although many board games could be suitable as a teaching method for computational thinking skills, we opted to design a board game inspired by parity games. In addition to potentially teaching computational thinking skills (such as problem decomposition, pattern recognition and abstraction), parity games teach commonly used graph theory concepts such as reachability and cycle detection. Furthermore, parity games have the interesting and rather unique property of having a single token which is controlled by two players. This token has to be steered towards favorable areas of the board while steering it away from unfavorable areas. Players will have to deal with these unfavorable areas by going around them, altering them to be more favorable or find a way to quickly cut through that area. As such, this new board game could offer a very interesting, new and unique gameplay experience.

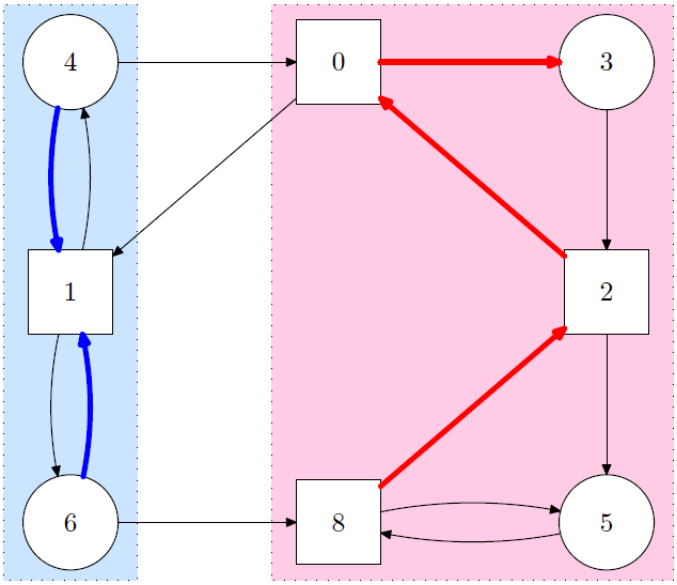

Figure 3.1: An example of a parity game [18].

The design of the game is inspired by parity games. So we will briefly discuss parity games before diving into the design of the game. A parity game is played by two players on a directed graph. Each node is associated with a player (the owner of that node) and a number. The players move a token along the nodes of the graph. The owner of the node the token is on decides where the token is moved next, it may be moved to any successor. The game ends when a player can force the token into an infinite cycle. Player one wins if the largest number that occurs in this cycle is even, otherwise player two wins. An example of a parity game is shown in figure 3.1. In this example, player one controls the token in the round positions and wins if the highest number on an infinite cycle is even. Player two controls the token in square positions and wins if the highest number is odd. In the blue area, player one can force a winning infinite cycle and in the red area, player two can force a winning infinite cycle.

3.2 Methodology

The game design process starts by setting up a set of requirements we believe the game should fulfill based on the preliminary study. Based on these requirements we create an initial design. This design is an adaptation of parity games. We make a paper prototype of the initial design and improve it over the course of three iterations of playtesting with colleagues, friends and family. During playtesting we experiment with different board layouts, cards, decks and rules. After the third round of playtesting we select the design we believe is most suitable and create 5 copies of the game for the school pilot. The simulation will also use the same design.

3.3 Requirements

We discuss a number of requirements below. For each requirement it is considered why it is important, how to measure if the game fulfills the requirement and what can be done if it does not:

1. **Primary school students (around age 9) should be able to learn the rules of the game in around 10 minutes**. This is important to ensure the school pilot sessions have a reasonable duration, so they do not take up too much of the students and teachers time. This can be tested by explaining the game to anyone (also someone older than 9), to see that it takes less than 10 minutes. If it takes much longer than 10 minutes, obsolete or unimportant rules should be removed from the game to simplify it or the explanation should be improved.
2. **The game should present the students with a complex system that requires them to engage systems thinking to understand all the relationships, levels and elements of the game**. This is important as it is part of the research goal to make a game that improves these skills. This is measured by asking the students how complex they think the game is. If it turns out the game is very easy to understand for them, more gameplay elements could be introduced (as long as they are meaningful and also improve the game in more ways than just adding complexity).
3. As it is part of the research goal, **the game should require mathemetical thinking skills like counting, probability, arithmetic**. At the end of each school pilot session, the students will be verbally asked about what strategies they used. At that point we can discuss how to improve the game to make the players more likely to engage these skills.
4. Also, as part of the research goal, **the game should require computational thinking skills like abstraction and problem decomposition**. It is difficult to test with young students if they actually use these skills, since they would not understand the question because they are too young to grasp these concepts.

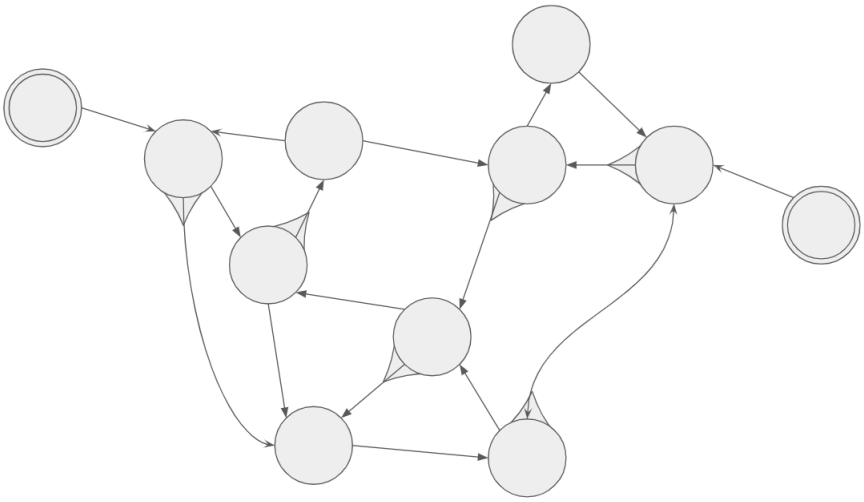

Figure 3.2: An example board

5. **The game should not last too long or too short**. The game should be long enough to allow the players to come up with interesting strategies to use and to let the advantage swing back and forth between the players. At the same time, the advantage of a short game is that players do not have to sit through a losing game for too long if they make a big mistake and that they can quickly try to deploy new strategies they came up with during a game. In the initial design, a game consists of three rounds. The aim is to have rounds that take around 5 minutes in a game that lasts two or three rounds. The duration of a round can be changed by increasing or reducing the number of cards in the deck. The number of rounds in a game can also be adjusted.

6. **The game should have enough depth, strategy and counterplay to stay interesting enough to replay many times**, so that the same game can be used for a long time by the same students. This will be tested by repeatedly playing the game. We know if the requirements are met if the game is still challenging and interesting even for experienced players. In addition, model checking could be used to simulate random games. If it turns out that there are a lot of simulations where the player who plays last or first wins by always playing a specific card at a specific time (for example the start or the end of each round), that means there is probably a relatively simple and powerful strategy that needs to be toned down by adding some restrictions to the game or tuning the numbers on the cards.

7. A snowball effect is when you can easily leverage your advantage in a game to increase your advantage. **The game should not snowball out of control** like this, because it is frustrating to be on the losing side and to be unable to recover. This requirement is confirmed by playtesting. If a snowball effect is discovered, a possible solution is to give the player who is behind some kind of edge in the next round (for example, they may decide the starting position of the figure or they may decide who goes first). We also discuss how this requirement can be tested using the simulation in section 7.3 on page 58.

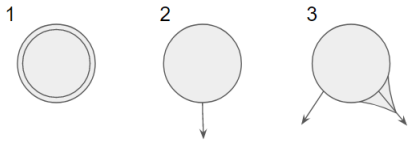

Figure 3.3: The three types of board positions. 1: Potential starting position. 2: A position with no choice where to go next. 3: A position with two choices where to go next.

3.4 Initial Design

In this section we describe the initial design of the game. This design is still quite open-ended as it has a lot of room for variations to try during play testing and game design iterations. Before diving into the details we give a brief overview of the flow of the game:

The game is played on a board with cards, a single figure, arrows and coins (each player has their own coin color). The goal of the game is to collect the most coins. One game consists of a number of rounds (for example, three, this number can be changed if the game takes too long or not long enough). During each round, players take turns playing cards which let them place coins on the board, change the direction of the arrows to program the figure's movements and determine the starting position for the figure. At the end of each round, the figure is moved across the board from a starting position until it has reached a position it has encountered before. While the figure moves, it collect the coins it encounters. The player who had the most coins on the route of the figure wins them all. Figure 3.2 on page 15 gives an example board. The edges between the positions are directional movement options (they can only be taken in one direction, indicated by the arrowhead). Each board position with multiple outgoing edges has an arrow that can be rotated to point at one of the outgoing edges. The figure will always move in that direction when it encounters this board position.

There are three types of board positions. They are depicted in figure 3.3 on page 16 and described in more detail below:

1. Potential initial position. Every board has at least one potential initial position. During each round, one of the players will place the figure on one of these positions. Players may not place coins here.
2. Position with a single possible outgoing edge. When the figure arrives here, it always goes in the only possible next direction.
3. Position with multiple possible outgoing directions. When the figure arrives here, it goes in the direction the arrow is pointing to.

3.4.1 Rounds and Cards

To create an element of randomness and hidden information, there will be a deck of cards. At the start of each round the deck is shuffled and every player is dealt 5 cards, this is the players hand, which will never have more than 5 cards. Players do not show each other their hand. Players take turns playing a card and drawing a card until the deck and each players hand is empty, after which the score is counted.

These are the cards in the deck:

- Place the figure on a starting position. There is only one copy of this card. A possible variation (if this card turns out to be too powerful) is to label each starting position with a

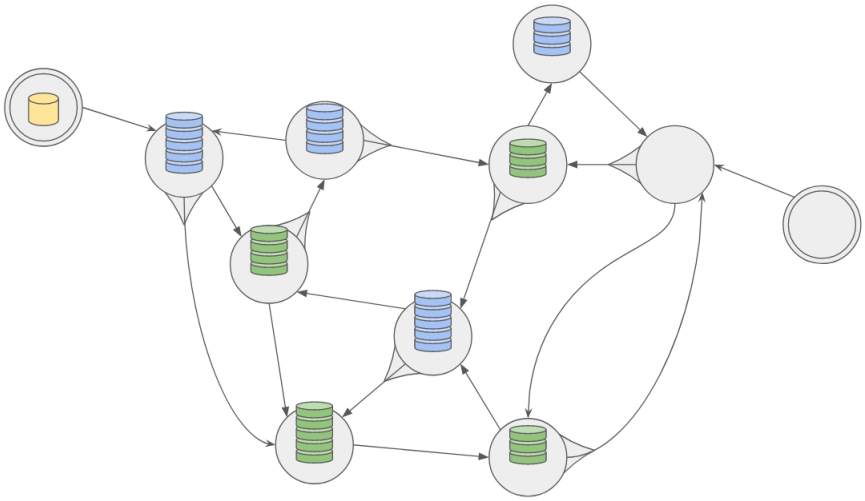

Figure 3.4: An example of the board after a round

number to indicate for which rounds it will be used. For example, if a starting position is marked ”1,3” it will always be used as the starting position in the first and third round.

- Change the direction of a board position. This card might be too powerful and create very ’swingy’ games (where the advantage swings from one player to another player too much) where it is difficult to find a meaningful strategy aside from using this card near the end of the round. Possible variations are to restrict this card so it can only be used once per board position (there would need to be exactly the right amount of copies of this card in the deck for the board that is being used) or restrict it so it can only be used on a specific position (for example, you can only use it on a position with your coins on it or no coins on it).
- Distribute 2 coins among 2 positions (place one coin on two positions or two coins on one position).
- Place 3 coins on a single position.
- Place 4 coins on a single position.
- Switch the coins between 2 positions.

When placing coins, the players may only choose a position that does not have coins placed by an opponent. Figure 3.4 on page 17 gives an example of how the board might look after two players (blue and green) are done with the first round.

3.4.2 Counting Score

When each player’s hand as well as the deck are empty, the score is counted. The figure is moved from the initial position. When it encounters a position with a single outgoing edge, it will go in that direction. When it encounters a position with multiple outgoing edges, it will take the direction that was ’programmed’ by the players when they pointed the arrow of that position in one of the directions. That means this process would be completely deterministic (players can not influence how the figure moves once the round ends). If deemed necessary, players could later be given control over the figure while it is moving. For example, a card could be added to the deck

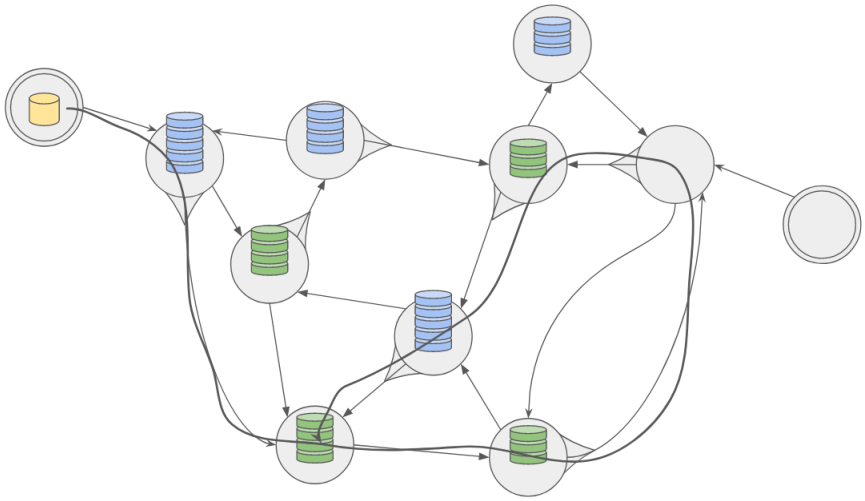

Figure 3.5: The figure moves until it has completed a cycle

that lets the player place a special figure on a position with multiple outgoing directions. The first time this special figure is encountered, the player determines the direction for that position. Figure 3.5 shows how the figure moves on the example board.

There are currently two methods in consideration for determining the 'winner' of a cycle:

- The player with the greatest *total* number of coins on the cycle wins all the coins on the cycle.
- The player with the largest number of coins on a *single* position on the cycle wins all the coins on the cycle.

Outside of the cycle, coins encountered are awarded to the player who placed those coins. In the case that the first option is chosen (the greatest total number of coins on the cycle), the score would be counted as shown in figure 3.6 on page 19. After counting the score, the board can be reset for the next round by removing the coins or the remaining coins can be left there (this is a game design choice). In the example (figure 3.6), if the coins would be left, the green player would have more points, but the blue player would enter the second round with more coins on the board. This also allows the players to use long term strategies which span multiple rounds.

3.5 Playtesting

After the first design phase we playtest the game. During this stage, the main goal is to test and improve gameplay aspects such as the size and layout of the board, the type of cards that are in the deck and their ratios. In addition, some variations of the game are tested, such as the win condition (the highest total number of coins a player has on the cycle versus the highest stack of coins a player has on the cycle).

3.5.1 First Iteration

We first started testing the game with the board from the initial design (see figure 3.8 on page 20) and the following deck:

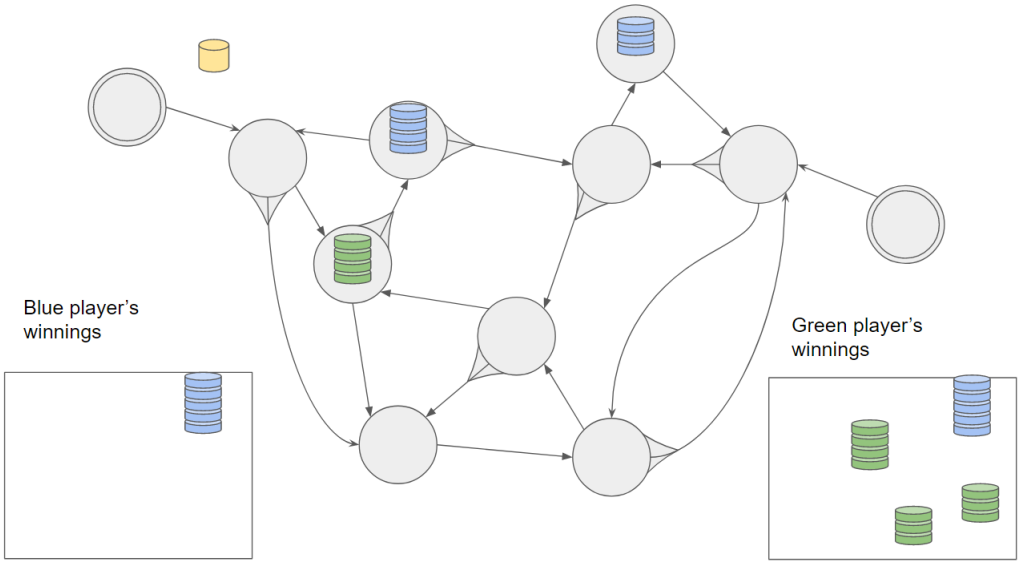

Figure 3.6: The score is counted

Figure 3.7: The finished paper prototype in action (during the second iteration of playtesting)

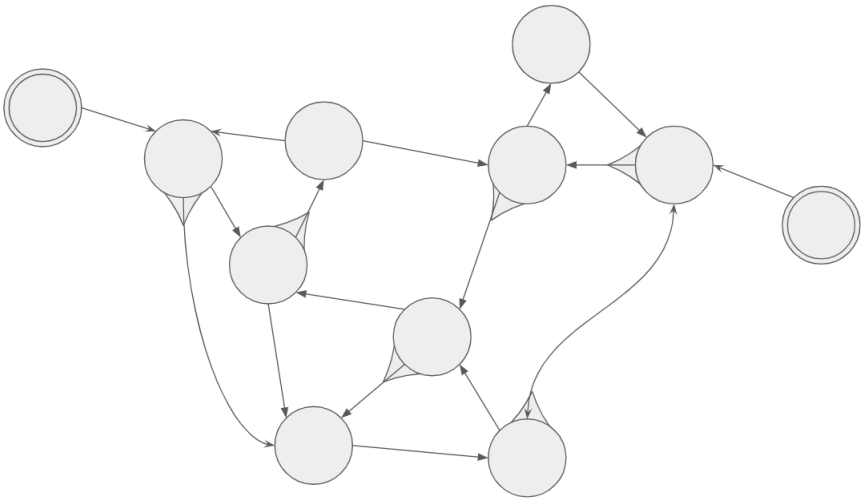

Figure 3.8: The board for the first prototype

#	Card
1	Place the figure on a starting position
9	Place 3 coins on a position
6	Distribute 2 coins (can place two on one position or one on two)
2	Switch coins between two positions (can use an empty position, can switch the coins of two different players)
7	Place an arrow and give it a direction
4	Change an arrow's direction
29	total cards

After playing the game several times we made the following observations:

- Playing the game with the rule "the player with the *single highest stack of coins* on the cycle wins all the coins in the cycle" was not fun at all. The whole game revolved around making one or two big stacks of coins and trying to switch them onto the cycle in the figure's path in the last few turns. The rest of the games in this iteration were played with the alternative option: "the player with the *greatest total number of coins* on the cycle wins all the coins in the cycle".
- One round of the game lasts around 15 minutes, which is too long.
- The board is too small for the amount of cards in the deck. As a consequence, the board fills up quickly with coins and a large portion of the game is played with no empty spaces on the board.
- Securing strong board positions (by placing just one coin) early on is very important.
- Saving up cards to switch coins and place/change an arrow is a powerful strategy as the winner is mostly decided in the last few turns of the game.
- After most of the strong board positions have been secured, the game is relatively uninteresting until the last few turns. The players are just spreading out their coins over the board positions they have until it starts to become clear how the figure is going to move.

- It is possible to hold the card ”place an arrow” and force other players who do not have this card to use their ”change an arrow’s direction” cards on different arrows. Additionally, the fact that there need to be as many copies of this card as there are board positions that can have arrows is problematic. When increasing or reducing the number of arrows on the board, the deck size and ratio of cards in the deck also changes.
- Whether the number of cards in the deck is odd or even is important. If the number of cards in the deck is odd, the player who goes first has a big advantage as they have first pick on which board positions they want to secure and they also get the last turn, which is a great opportunity to impact the route of the game figure. Ideally, the deck size is even so that the player who goes first does not get the last turn.
- It is unintuitive that arrows can only point in the direction of certain edges. The directionality of the edges should be made more obvious.
- The fact that only the coins on the cycle are counted for the winning player (instead of all coins encountered) is unintuitive and not very impactful for the gameplay.

3.5.2 Second Iteration

After the first round of playtesting, we have made the following changes to the game:

- Instead of winning only the coins on the final cycle, the round winner gets all the coins encountered by the game figure.
- The card ”place an arrow and give it a direction” has been scrapped. Instead, positions that can have arrows will now always have an arrow at the start of the game. On the board, a default direction for each arrow is indicated.
- The board has been expanded with a second ’cluster’ of positions. The original board has been altered somewhat to keep the board as a whole balanced, although it still has the general layout it had before (thee groups of three positions).
- The overall number of cards has been reduced to make the games quicker and hopefully remove the middle part of the round where the choices players make are almost irrelevant.
- The number of rounds has been reduced from 3 to 2.
- To make up for the lack of cards, a rule has been added. After playing a card, the player places an additional coin on any position they already have at least one coin.
- Each card now mentions how many copies of that card are in the deck.

This resulted in the board shown in figure 3.9 on page 22 and the following deck:

#	Card
1	Place the figure on a starting position
7	Place 3 coins on a position
4	Distribute 2 coins (can place two on one position or one on two)
2	Switch coins between two positions (can use an empty position, can switch the coins of two different players)
6	Change an arrow’s direction
20	total cards

After playing with the second paper prototype we made the following observations:

- The rounds now take around 10 minutes, which seems like a good duration.

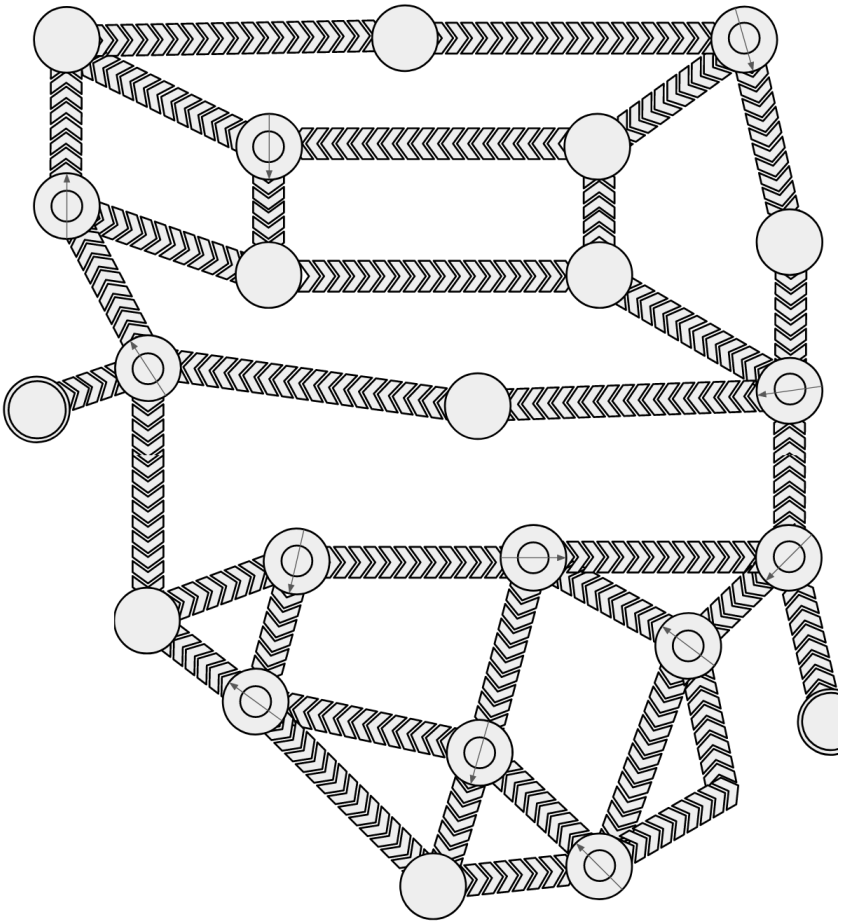

Figure 3.9: The board for the second prototype

- Some of the players were skeptical because of how the paper prototype looked, but ultimately said the game was fun.
- One player said she thinks the game is likely appealing to the same people who would enjoy playing chess.
- The number of cards was quite low when playing the game with 3 players (only 3 cards in the deck at the start of the game, after dealing each player a hand).
- The 'old' section of the board looks too busy and did not get used as much as the new section.
- The card 'place the figure on a starting position' was making the game too unpredictable.

3.5.3 Third Iteration

Based on the observations from the second iteration we have made the following changes:

- Each turn, players place a coin *or remove someone's coin* from the board (instead of always just placing a coin). This allows players to take back positions from other players if they do not invest enough coins into it.
- The card 'switch coins between two positions' is too powerful, and is changed to 'move coins from one position to another'.
- Players can no longer place coins on switches.
- There is now a different amount of cards for 2 and 3 player games. Two player games now take two rounds and three player games take three rounds.
- The card 'place the figure on the starting position' has been scrapped. Instead, the train now starts at the first station in the first round and then alternates between the two possible starting stations.
- Since the starting position of the train is now set, we no longer count the coins encountered before the final cycle.

The first column in the table below shows how many copies there are of each card for 2 player games. The second column show how many there are for three player games.

#2	#3	Card
1	1	Place the figure on a starting position
3	4	Place 3 coins on a position
4	5	Distribute 2 coins (can place two on one position or one on two)
2	3	move coins
4	5	Change an arrow's direction
14	18	total cards

With this prototype we made the following observations:

- The deck is still too small for three player games.
- Having less locations to place coins is good. It is now more important where players place coins in the early stage of the game.
- Adult players took a long amount of time to analyze the last few turns of the game, leading to long games.

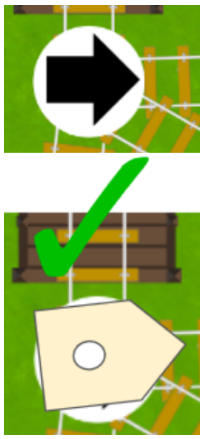

Figure 3.10: Lift up the wooden arrow to reveal the default switch direction

3.6 Final game rules

The game rules are divided into five sections: the overview, the turn structure, the round start, the cards and the round end.

3.6.1 Overview

The game can be played with 2 or 3 players. For 3 players, extra cards are put into the deck (see subsection 3.6.4 'Cards'). In a 2 player game, the players play 2 rounds. In a 3 player game, the players play 3 rounds. Players decide who goes first and then take turns in a clockwise fashion. Once all the cards have been played, the round ends. The next round, the next player (clockwise) takes the first turn. The same goes for the third round (if there is one). This way, everyone gets a chance to play first. There are two starting stations ('eerste beginstation' and 'tweede beginstation'). In the first round, the train starts on the first starting station. In the second round, the train starts on the second starting station. If there is a third round, the first starting station is used again.

3.6.2 Turn Structure

Each turn consists of 3 parts. These actions must all be taken, in the specified order:

- Place or remove a coin. You can either place a coin of your own color or remove a coin belonging to another player from the board entirely. You may only place or remove a single coin. You may only place a coin on a station with no coins or on a station with coins of the same color. You may remove a single coin from a station with more than one coins.
- Play a card. The card goes on the played cards pile, face up. Follow the instructions on the card.
- If possible, draw a new card from the face down card pile. Do not draw cards from the face up played cards pile.

3.6.3 Round Start

At the start of each round:

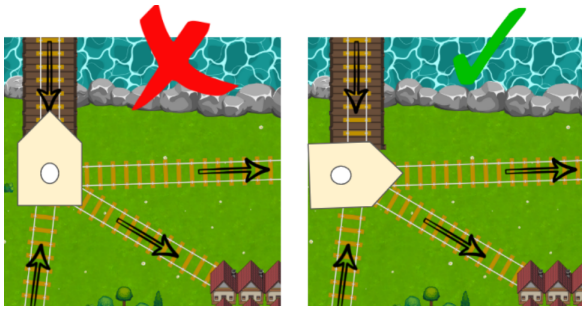

Figure 3.11: On the left, an invalid switch setting. On the right, a valid switch setting

- Make sure the deck is correct (see the 'Cards' subsection). For 2 players, there should be 16 cards. For 3 players, there should be 21 cards.
- Shuffle the cards and deal each player a hand of five cards. Players can look at their own cards, but not at the cards of other players. Place the rest of the cards in a face down pile.
- Remove any coins that are still on the board.
- Put all the switches in the default position (see figure 3.10).
- Place the train figure on the appropriate starting station.

3.6.4 Cards

Below is a list of the 4 cards with an explanation of how they work and how many copies there are in the deck for 2 and 3 player games. For 2 players, the deck has 16 cards. This means that one player takes the first turn and has the advantage of being able to place coins on any station, but the other player gets to play the last card, so they get a final say before the score is counted for that round. The next round, the roles are reversed. For 3 players, the deck has 21 cards. This means that one player goes first, another player goes last and one player gets neither advantage. Because of this, it is important to rotate who gets to play first and to play the correct amount of rounds (2 for 2 players and 3 for 3 players). That way every player gets the advantage of playing the first and last cards just as often.

- 'Plaats 3 muntjes' (place 3 coins). This card lets you place 3 coins on a single station. You are only allowed to place coins on a station with no coins from another player. 2 players: 5 copies, 3 players: 6 copies.
- 'Verplaats muntjes' (move coins). This card lets you move all the coins from one station to another station. You can move the coins from any station you want (even with coins from another player), but you may only move them to a station with no coins or coins of the same color. 2 players: 2 copies, 3 players: 4 copies.
- 'Verdeel 2 muntjes' (distribute 2 coins). This card lets you distribute 2 coins among 2 staitons. So you can either place 2 coins on 1 station or 1 coin on 2 stations. Once again, you may only place coins on a station with no coins or a station with coins of the same color. 2 players: 5 copies, 3 players: 6 copies.
- 'Verander de richting van een wissel' (change the direction of a switch). This card lets you change the direction of one of the switches, as long as it is not sending the train onto the track in the wrong direction (see figure 3.11). 2 players: 4 copies, 3 players: 5 copies.

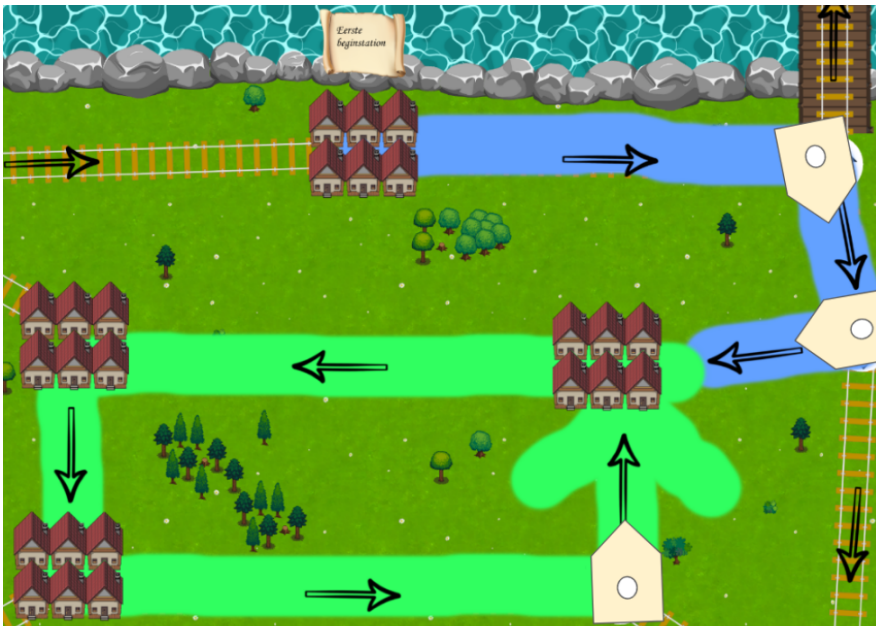

Figure 3.12: An example of train movement. Only the coins on green section are counted

3.6.5 Round End

Once all the cards have been played (the face down card pile is empty and the players no longer have any cards in hand) the round has ended and the score is counted. To count the score, move the train from the starting station, following the directions of the tracks and the switches. When the train encounters a station it has encountered before, the train has encountered a cycle. This cycle is crucial as only the coins on this cycle are counted. An example of this is shown in figure 3.12 on page 26. Another crucial rule is that *the player with the most coins on this cycle, wins all the coins on this cycle*. Once the score has been determined, write it down so the next round can begin or determine the winner if this was the final round.

3.7 Final Game Design

In order to easily allow for future changes and expansions, we opted for a modular design as outlined in figure 3.14 on page 27. It is possible to make a completely different board by just switching the thin covers. The solid board underneath can have many holes, so you only need one set of boards to play with many different covers. It is also possible to make larger or smaller board layouts by using a different amount of boards. If the tracks leave and enter the edge of the covers at set points, the players can customize the layout by combining many different covers.

Due to time constraints and to ensure the experience of the game is the same for every student in the school pilot, we only made a graphical design for one set of covers, see figure 3.13 on page 27. We made 5 copies of the game using wood for the solid boards and thick Colotech paper printed at Xerox for the covers and cards. The cards were also laminated so they will last longer. The arrows for the switches are made of wood and metal pins were used to keep everything together. Sets of bingo coins and train figures were purchased online. Small pieces of felt were placed on the corners of the underside of the wooden boards so they do not damage the surface they are placed

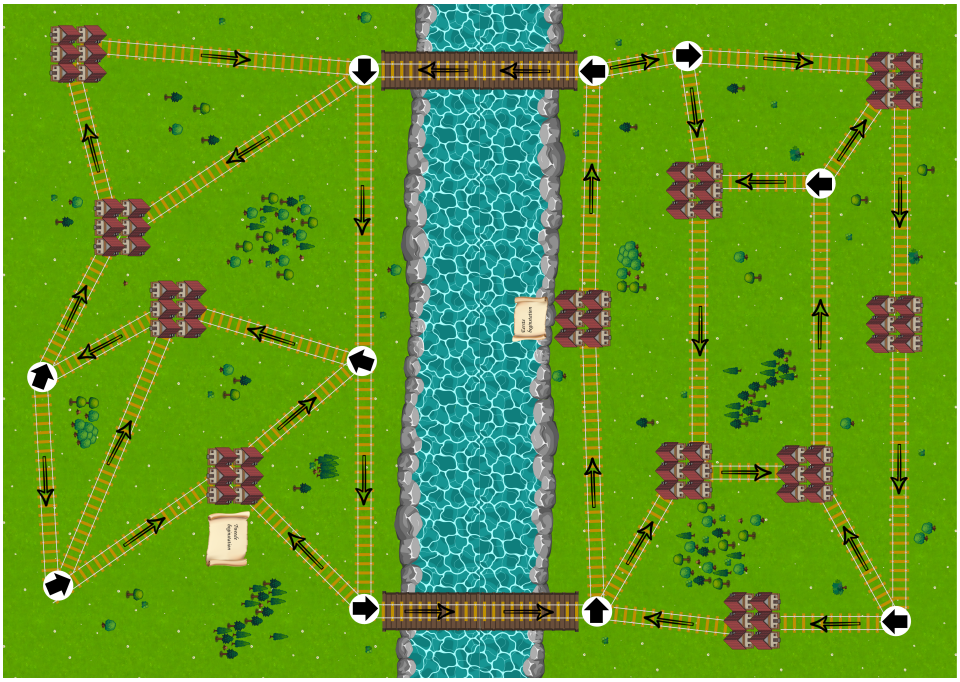

Figure 3.13: The graphical design for the board cover

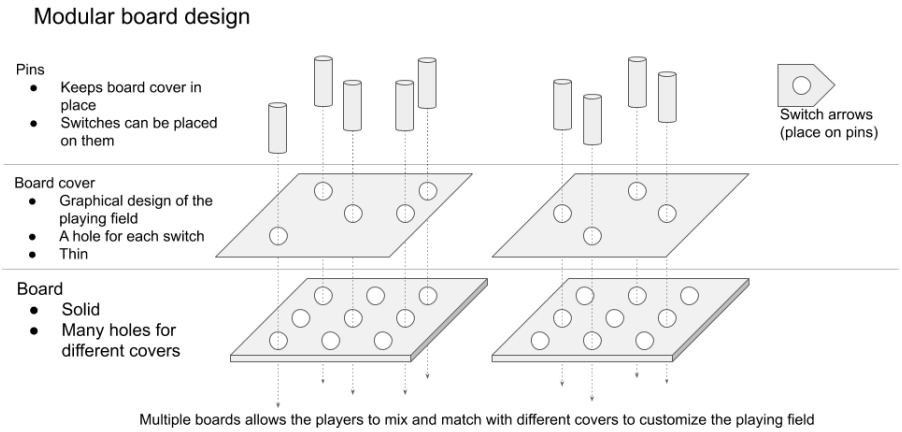

Figure 3.14: Concept for board game design

Figure 3.15: The finished board game

on. The final result can be seen in figure 3.15 on page 27. One of the copies of the game resides at the university, 3 with the schools that participated in the school pilot and one with the researcher.

3.8 Thinking Styles and Strategies in the Proposed Game

The game is related to computational thinking, mathematical thinking activities and graph theory in a number of ways:

- Algorithms: the figure moves along a path which is 'programmed' by the players. There are cards that let the players determine how the figure will move when it encounters a board position with multiple possible movement options. This results in a completely deterministic path the figure follows at the end of each round. This path can be seen as a program or algorithm created by the players. In a way, this teaches the basic concept behind a program or algorithm: something is following a set of commands. In this case, the players are all trying to change the set of commands for the same figure to their own benefit.
- Cycle detection: finding cycles in a graph is a common exercise in graph theory. Creating an algorithm to do this is also an exercise that is sometimes used in programming courses. Finding and using such a systematic way to find cycles becomes increasingly important for the players as they move to more advanced boards which could have dozens of cycles. This could be something as simple as mentally iterating over each board position and considering all the possible cycles that position is part of.
- Abstraction: abstraction is removing unnecessary details (such as the exact amount of coins or the specific layout of the board) and to instead mentally mark an area as favorable or unfavorable. This allows the player to steer the figure towards favorable areas and away from unfavorable areas without constantly doing the math of how many coins would be won by everybody for several different paths.
- Problem decomposition: the 'problem' of winning the game needs to be decomposed into smaller problems such as "how to deal with a problematic area on the board created by the

opponent", "how to best utilize the cards in hand", "which areas of the board are important to invest in", etc.

- Systems Thinking: especially for the young students in the pilot, the game will present them with a complex system they need to analyze and understand, with all its levels and relationships.
- Mathematical Thinking: the math skills involved while playing the game are counting, arithmetics and probability. Counting and arithmetics are important for example for determining the winner of a particular cycle or calculating how many coins still need to be won in a round to win the game. Probability is used when reasoning about the cards in the hands of the other players.
- Modeling and simulation practises (from Weintrop's computational thinking definition [16]): although the players usually will not have any computational modelling or simulating tools at hand while playing the game, they will still mentally create a conceptual model and attempt to do some simulations of how the figure will move in their head. This model is used to make decisions about the game.

In addition, there are a number of strategies or lines of play the players could try to use:

- General board assessment: identifying strong board positions and locking them down early by placing coins on them.
- Reducing risk by spreading out coins instead of placing them all on a single board position.
- Identifying strategies and counterplay: Disrupting an opponent's strategy by, for example, placing a small amount of coins on a key position for their assumed strategy.
- Playing to your outs: in the last round of a game, the player who is behind has only one way to have a chance to win the game and that is by winning all the coins on a large cycle. Even if this requires drawing certain cards from the deck to make an unlikely strategy come together, the player should still play as if they draw exactly the cards they need to make this happen. This is only valid in a situation where taking a safer line of play (trying to score a smaller cycle) would result in almost certain defeat.
- Hidden information & probability: reasoning about the cards in the other player's hands. Based on their previous plays they could have a certain strategy which hints to which cards they still have in hand. Based on knowledge of the deck and the number of each card played, the probability the opponent has certain cards goes up or down. For example, consider a three player game. All but two of the 'change direction' cards have been played, one is in your hand. Opponent A has placed a lot of coins on the path the figure is currently programmed for while opponent B has placed a lot of coins on a different path. This makes it likely that opponent B has the last 'change direction' card in his hand (or they are following a 'play to your outs' strategy and must draw the last 'change direction' card from the deck to have a chance at winning the game).

In conclusion, this is a game where we expect the players to use and improve several skills and aspects of computational, mathematical and systems thinking.

Chapter 4

School Pilot

To find out how suitable the game is for education, we took it to primary schools to test it in a school pilot. The goal of the pilot was to find out how well the students understood the game, how they played it, which strategies they used and what their opinion of the game was.

4.1 Participating Students

Three schools in a suburban area in the middle of the Netherlands were willing to participate in the study. The schools were all located close together. For each school, one grade was selected. All students of that grade that got permission from their parents to participate, participated. Figure 4.1, 4.2 and 4.3 visualize how the students were divided into groups of 2 or 3 who they played the game with along with their ages and genders. At the day of the pilot, for each school, the teacher send in groups of 5 to 8 students at a time. The students were asked if they had a preference for who to play the game with. If they did not have one, they were assigned randomly.

In total, 44 students participated. 23 were female and 21 were male. 1 student was 8 years old, 12 were 9 years old, 15 were 10 years old, 13 were 11 years old, 2 were 12 years old and one student failed to specify their age.

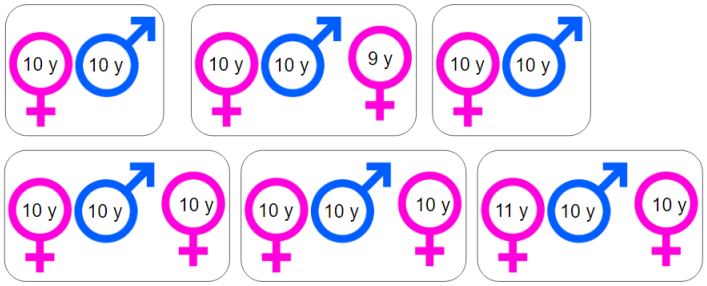

Figure 4.1: School 1: 'groep 7'

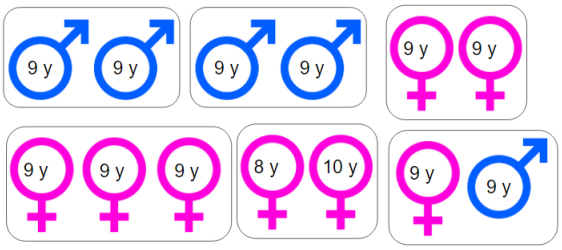

Figure 4.2: School 2: 'groep 6'

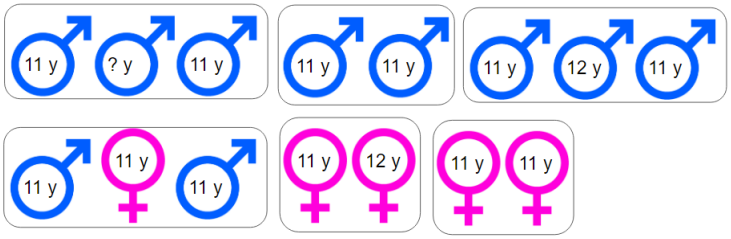

Figure 4.3: School 3: 'groep 8'

4.2 Methodology

The pilot happened in a separate classroom. The teacher would sometimes check in, aside from that it was just the main researcher (Steven) with the students. Students participated in the pilot in groups of 5 to 8 students at a time. Each group spend roughly 1 hour participating in the pilot. The first 15 minutes were used to explain how the game works, play a practise round (while observing how they played) and then correct any misunderstandings. Then they played the game for 30 minutes, during which the scores were recorded. In the last 15 minutes, the students filled out a survey. Because each group was scheduled for an hour and some students played faster than others, not every student played the same amount of rounds.

The data from the surveys was entered in a spreadsheet and then imported into a python script. In the python script, the data was split in a number of ways:

- Competitiveness: students rated how much they like to win. Not at all (0), a little (1) or a lot (2). The low competitiveness group consisted of 17 students (0) and the high competitiveness group consisted of 26 students (1 or 2).
- Age: how many years old. Since students from 3 different grades participated, the best possible split was not very even. There were 28 students in the low age group (8, 9 or 10 years old) and 15 students in the high age group (11 or 12 years old).
- Gender: all students identified themselves as male or female. There were 23 female students and 21 male students.
- Frequency of playing board games: students indicated how often they play board games. On a yearly basis or less (0), on a monthly basis (1) or on a weekly basis (2). There were 28 students in the low frequency group (0 or 1) and 16 students in the high frequency group (2).
- Total score: the total amount of points the student accumulated during the 30 minutes of playing (practise round not included). There were 23 students in the low score group (0 to

10 points) and 21 students in the high score group (12 to 69 points).

- Win or loss: if the student scored more points than the student(s) they played against, they won. Otherwise they lost. 19 students won and 25 students lost. There were more losers than winners because three player games only have one winner.
- Closeness of the game: In a two player game, the closeness of the game is the difference in total score between the two students. In a three player game, the closeness for a particular student is the average of the score differences with the two other students. This means that the student with the middle score of the three experiences a smaller difference in score with the two other students than the leading student or the student with the lowest score. We think this accurately reflects how close the game feels to each student. There were 23 students in the low score difference group (2.5 to 15 average difference in score) and 21 students in the high score difference group (17 to 61 average difference in score).
- Group size. Students played the game in groups of 2 or 3 students. 20 students played in groups of 2 and 24 students played in groups of three.

Some students simply did not answer every question. For example, one student did not specify their age and another did not indicate how competitive they were. These students were simply left out when splitting the samples on age or competitiveness.

Aside from the aforementioned factors (age, gender, etc) the survey measured the following:

- How fun the game was, rated from 0 to 4. A higher score means more fun.
- How much the students wanted to play the game in their free time, rated from 0 to 4. A higher score means a bigger desire to play the game.
- Complexity of the game, rated from 0 to 4. Higher means more complex.
- How exciting (or boring) the game is, rated from 0 to 4. Higher means more exciting, lower means more boring.
- How often it was difficult to pick a good move, rated from 0 to 4. Higher means more often.
- How difficult it was to come up with a strategy in general, rated from 0 to 4. Higher means more difficult.

The values from the first two questions (how fun the game was and how much they want to play it in their free time) are added together to form the enjoyment score (from 0 to 8). The values from the last two questions (how often was it difficult to pick a good move and how difficult was it to form a strategy in general) are added together to form the strategic difficulty score (from 0 to 8). These questions were chosen with the intention to do this, because there are multiple aspects to how enjoyable or strategically difficult the game is, but each individual question has to be understandable for the students.

With all of this data, we test the following hypotheses:

1. The game is less complex and/or less exciting for older students.
2. Boys and girls could very well develop their cognitive abilities at a different rate around this age. We will investigate if there can be found any differences at all for the measured data between boys and girls.
3. Competitive students are more eager to learn how the game works so they can win. As a result they will score more points and win more often.
4. Competitive students who win enjoy the game more than other students who win.

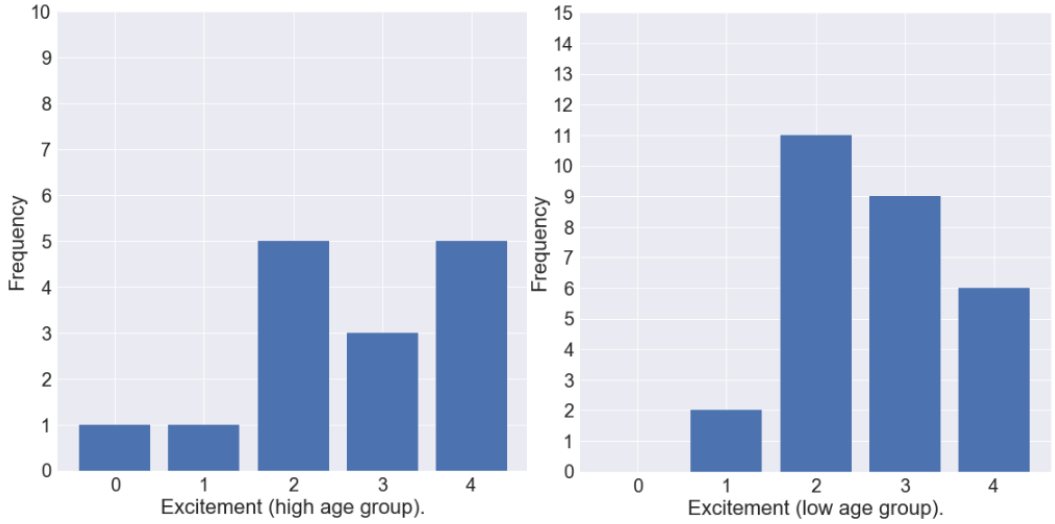

Figure 4.4: Excitement comparison for high and low age group.

5. Students who frequently play board games will find the game less complex.
6. Playing the game with a different group size (2 or 3 players) significantly changes the dynamic of the game. We will investigate if there can be found any differences at all for the measured data between 2 and 3 player games.
7. Students who win or score a lot of points enjoy the game more.
8. Students who have very close games (small score difference) enjoy the game more.
9. Students who thought the game was very complex enjoyed the game less.

To test these hypotheses, the data is split, visualized and tested in Python. For example for hypothesis 1, 'the game is less complex for older students', we split the data into a low age group and a high age group. Then we visualize the data by creating a bar graph for the complexity experienced for each group. Finally, the two complexity samples are compared with the Mann Whitney U test.

The Mann Whitney U test is used to test the vast majority of hypotheses. It is appropriate because it is unclear what the distribution of the data is and the data is in a small discrete range. When splitting the data in multiple ways, such as with the hypothesis that winning is more important for competitive students, we use a contingency table and the Chi-squared test. When comparing data in a larger range, such as the total points scored, we use the Wilcoxon rank sum test. The chosen alpha value is 0.05.

4.3 Data Visualization and Testing

4.3.1 Hypothesis: the game is less complex and/or less exciting for older students

We use the Mann Whitney U test for this hypothesis. The data is visualized in figure 4.4 and 4.5. For the excitement comparison, the p value is 0.238, so the null hypothesis that these samples are from the same distribution cannot be rejected. For the complexity comparison, the p value

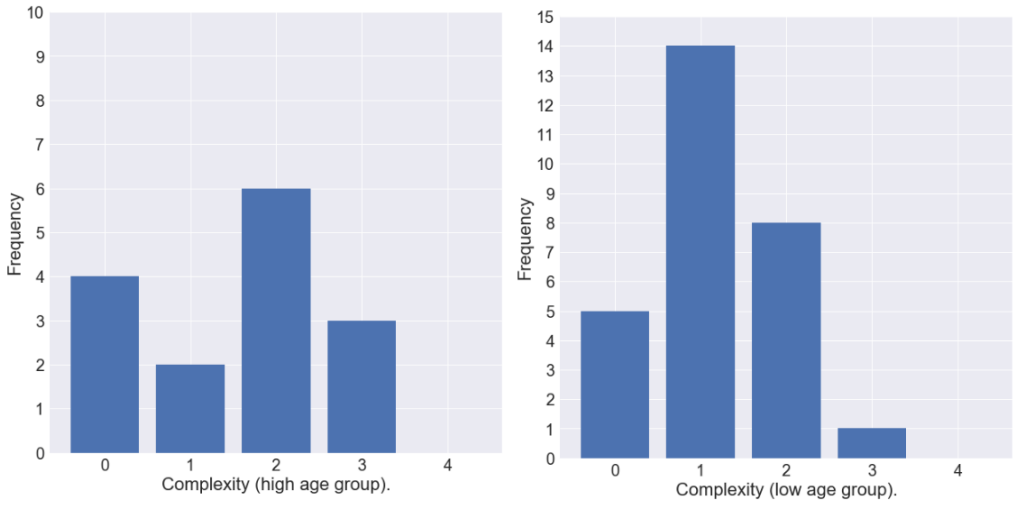

Figure 4.5: Complexity comparison for high and low age group.

is 0.883, so the null hypothesis that these samples are from the same distribution also cannot be rejected.

4.3.2 Hypothesis: there are differences in the experience based on gender

The data is visualized in figures 4.6, 4.7, 4.8 and 4.9 on page 35 and 36. For all the measured data, the Mann Whitney U test is used. The p values are:

- Complexity: 0.720
- Enjoyment: 0.716
- Excitement: 0.523
- Strategic difficulty: 0.885

None of the null hypotheses (that the samples are from a different distribution) could be rejected.

	high competitiveness	low competitiveness	total
winners	11	8	19
losers	15	9	24
total	26	17	43

Table 4.1: Contingency table showing the number of wins for above and below average competitiveness players

4.3.3 Hypothesis: competitive students perform better

Figure 4.10 shows the distribution plots for points scored by the low and high competitiveness groups. The samples are also tested with the Wilcoxon rank sum test, which resulted in a p value of 0.451, so the null hypothesis that these samples are from the same distribution cannot be rejected.

Table 4.1 on page 34 shows how many students fall into each category (high competitiveness winners, high competitiveness losers, etc). This data can be tested with the Chi-squared test.

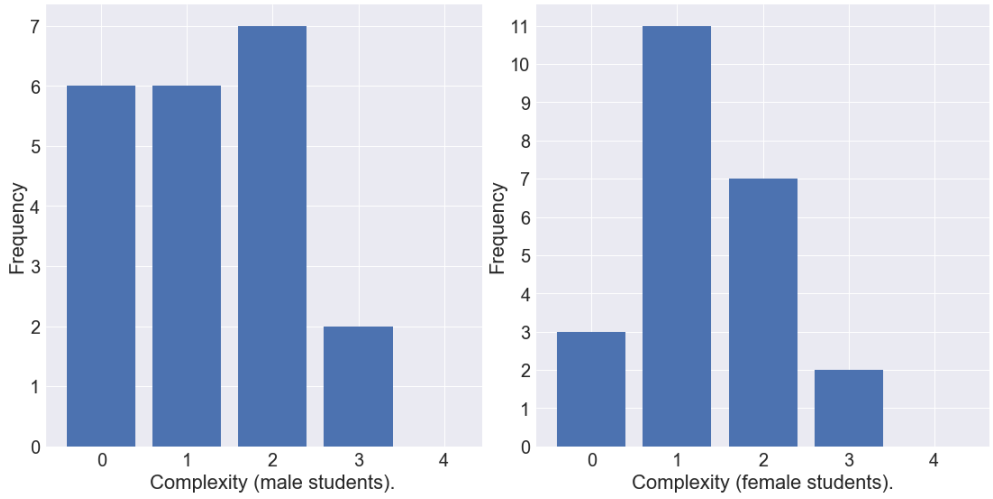

Figure 4.6: Complexity comparison for gender.

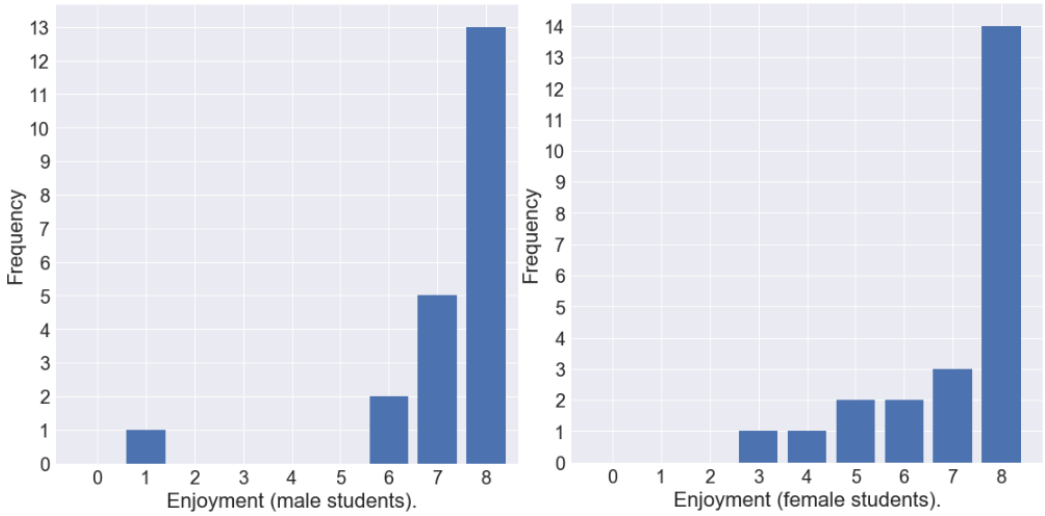

Figure 4.7: Enjoyment comparison for gender.

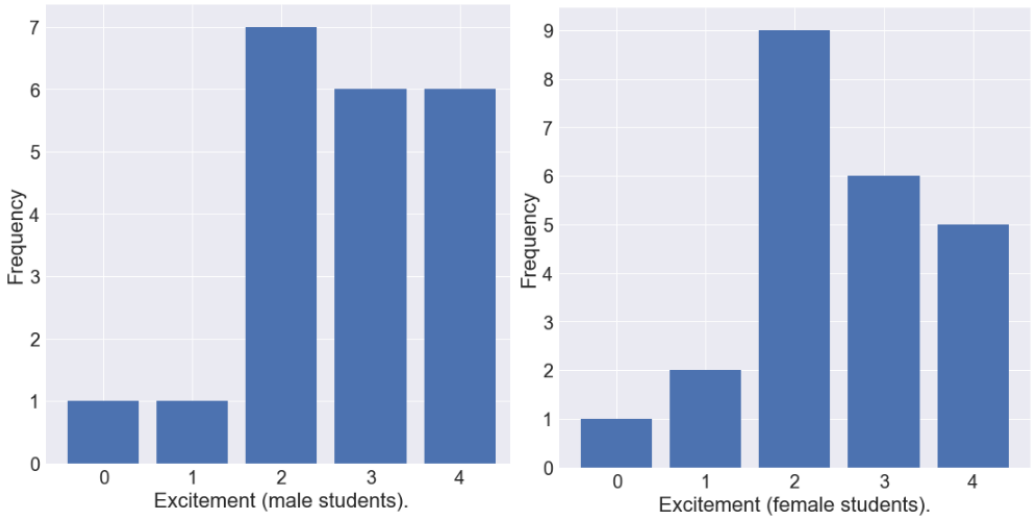

Figure 4.8: Excitement comparison for gender.

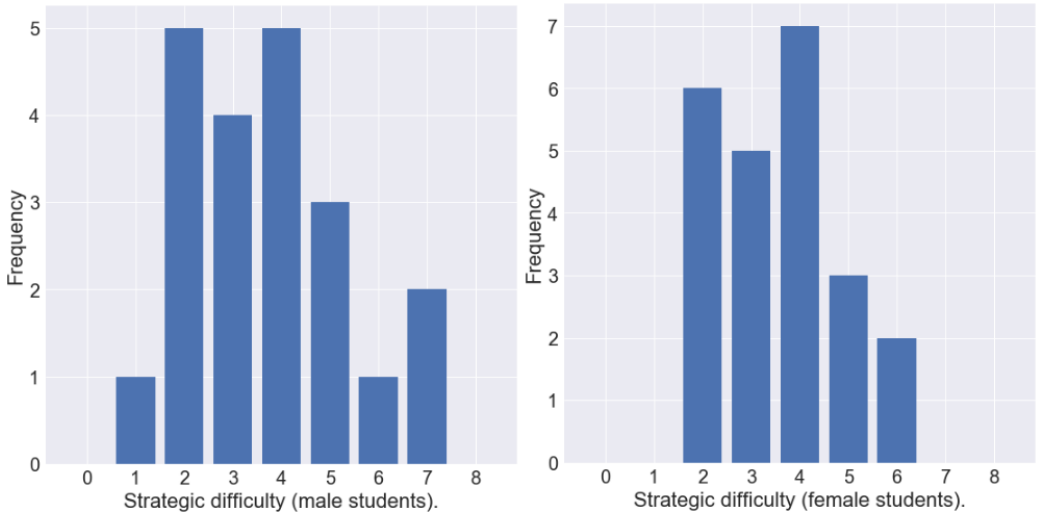

Figure 4.9: Strategic difficulty comparison for gender.

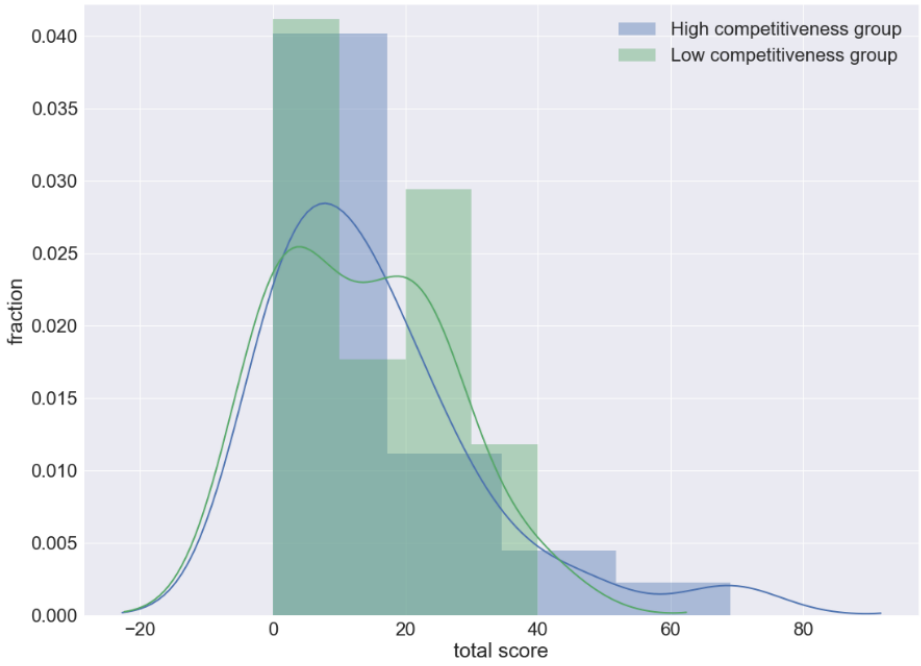

Figure 4.10: Total score comparison for competitiveness.

The expected frequencies used (based on the ratio of total winners and losers and high and low competitiveness students) are 11.5 for high competitiveness winners, 7.5 for low competitiveness winners, 14.5 for high competitiveness losers and 9.5 for low competitiveness losers. The null hypothesis is that the expected frequencies are accurate, which would mean the hypothesis is false. The p value from the test is 0.992. As such, the null hypothesis could not be rejected.

	high competitiveness	low competitiveness
winners	7.09	7.38
losers	7.00	7.11

Table 4.2: The average enjoyment of the game based on competitiveness and game outcome

4.3.4 Hypothesis: competitive students who win enjoy the game more than other students who win

Table 4.2 shows the enjoyment for the students in the different categories (competitive winners, competitive losers, etc). If anything, the difference in enjoyment between winners and losers is greater for less competitive students. Also, these samples are too small for proper statistical testing (N around 10).

4.3.5 Hypothesis: students who frequently play board games find the game less complex

The data for this hypothesis is visualized in figure 4.11 on page 38. The resulting p value from the Mann Whitney U test is 0.700 which means the null hypothesis could not be rejected.

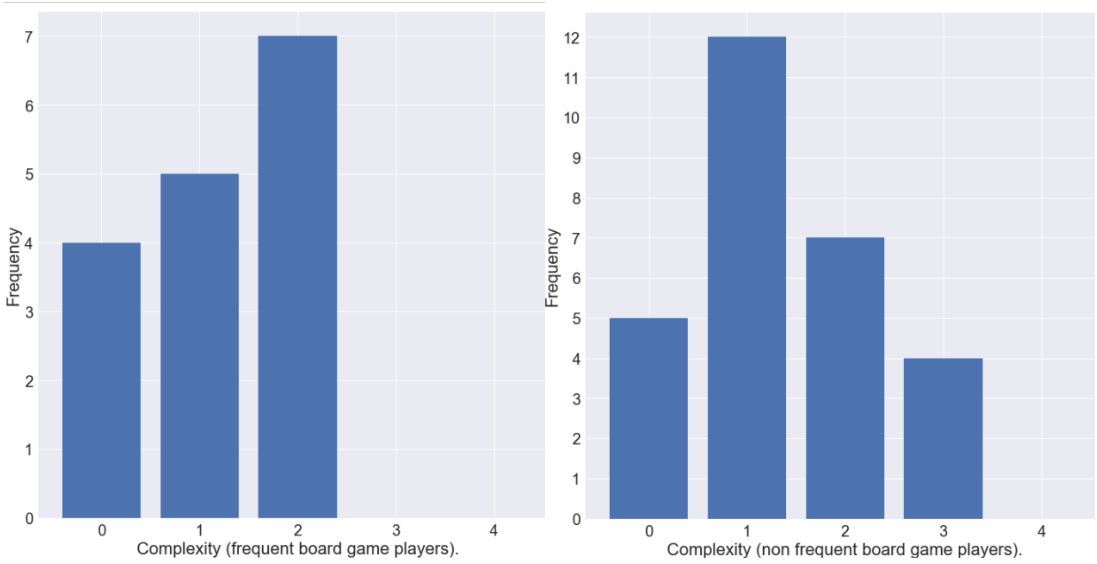

Figure 4.11: Complexity comparison for frequency of playing board games.

4.3.6 Hypothesis: there are differences in the experience based on group size

This hypothesis is tested for several metrics. The data is shown in figures 4.12, 4.13, 4.14 and 4.15. The Mann Whitney U test p values are:

- Complexity: 0.214
- Enjoyment: 0.181
- Excitement: 0.205
- Strategic difficulty: 0.385

None of the null hypotheses could be rejected.

4.3.7 Hypothesis: students who perform better enjoy the game more

We test this hypothesis for game outcome (figure 4.17 on page 41) and total score (figure 4.16 on page 41). The Mann Whitney U test results are 0.492 for the score comparison and 0.704 for the game outcome comparison.

4.3.8 Hypothesis: students enjoy the game more when there is a low difference in score

We tested the hypothesis by comparing excitement and enjoyment for the low and high score difference groups. The p value for the excitement comparison (figure 4.19 on page 42) is 0.244 and for the enjoyment comparison (figure 4.18 on page 42) it is 0.0221. There is evidence to reject the null hypothesis that students do not enjoy the game more when there is a low difference in score.

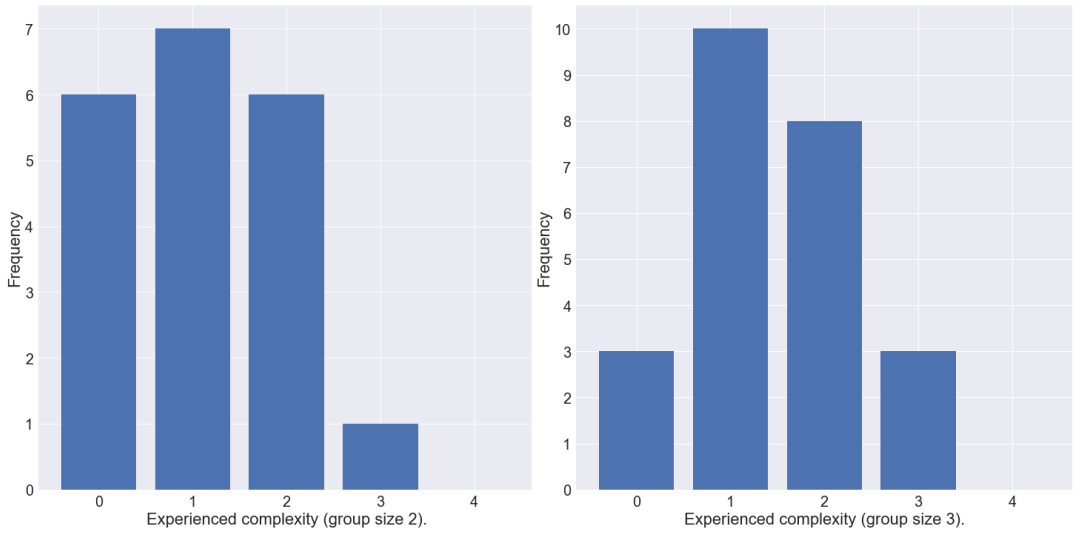

Figure 4.12: Complexity comparison for group size.

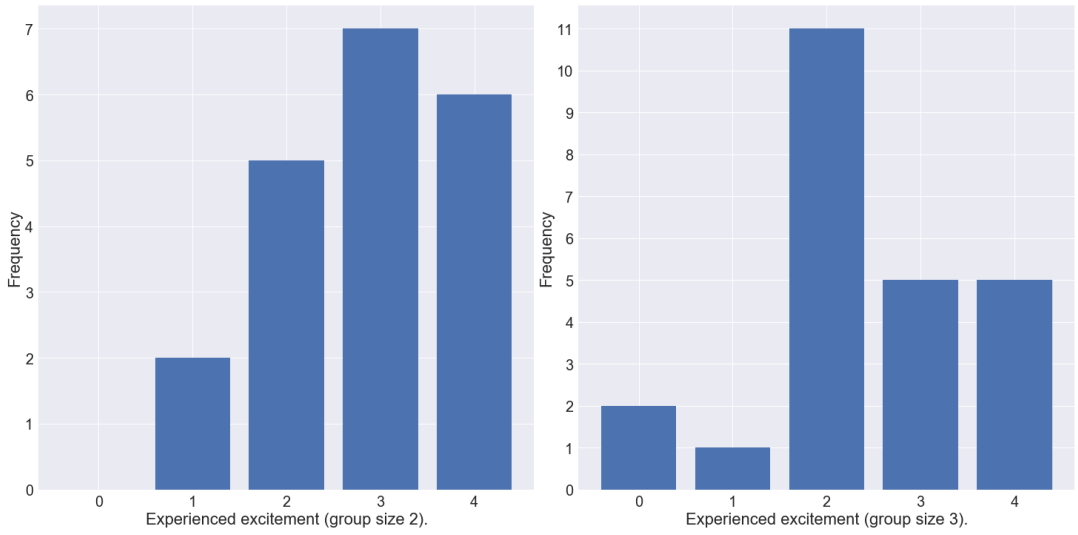

Figure 4.13: Excitement comparison for group size.

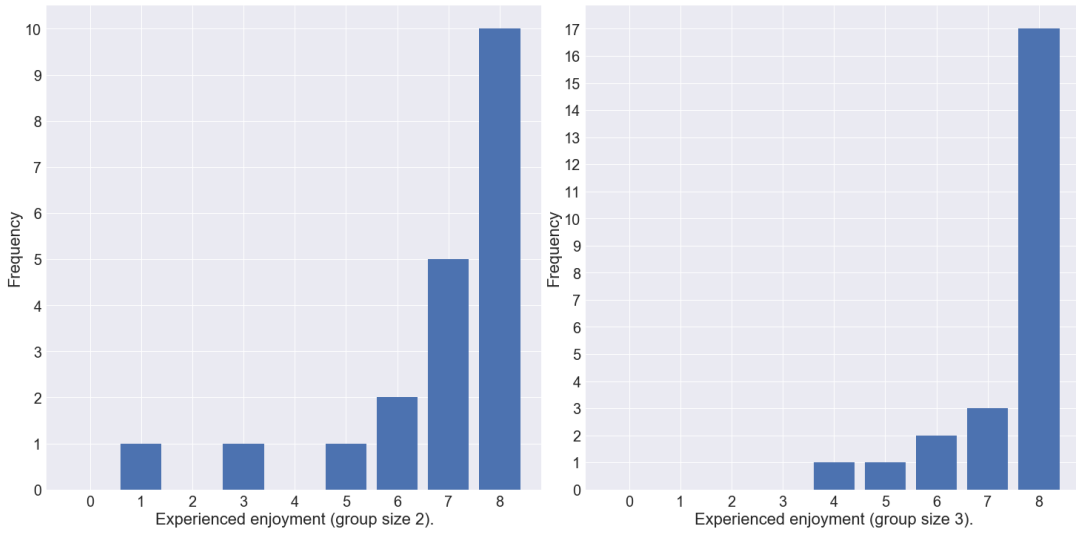

Figure 4.14: Enjoyment comparison for group size.

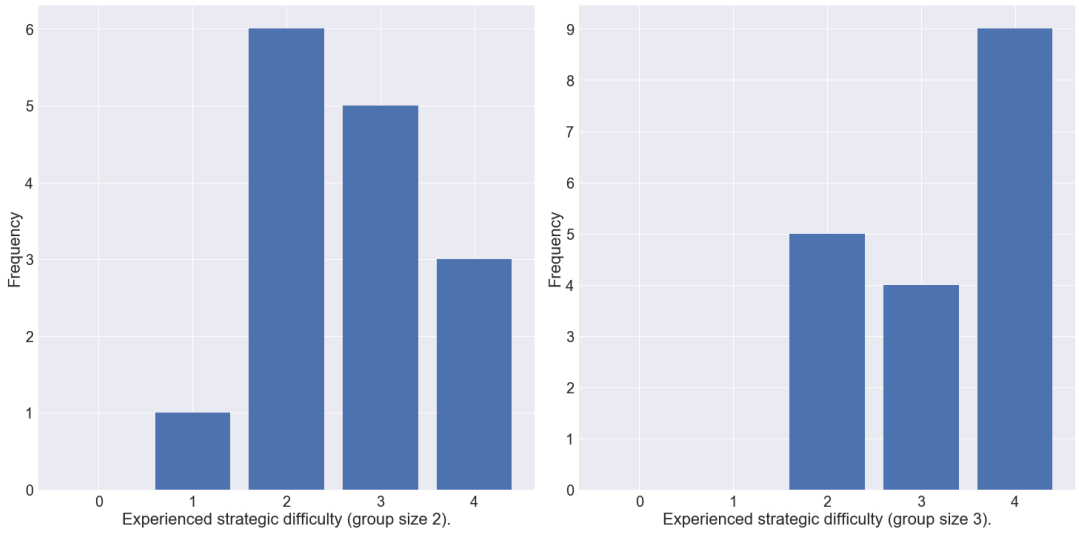

Figure 4.15: Strategic difficulty comparison for group size.

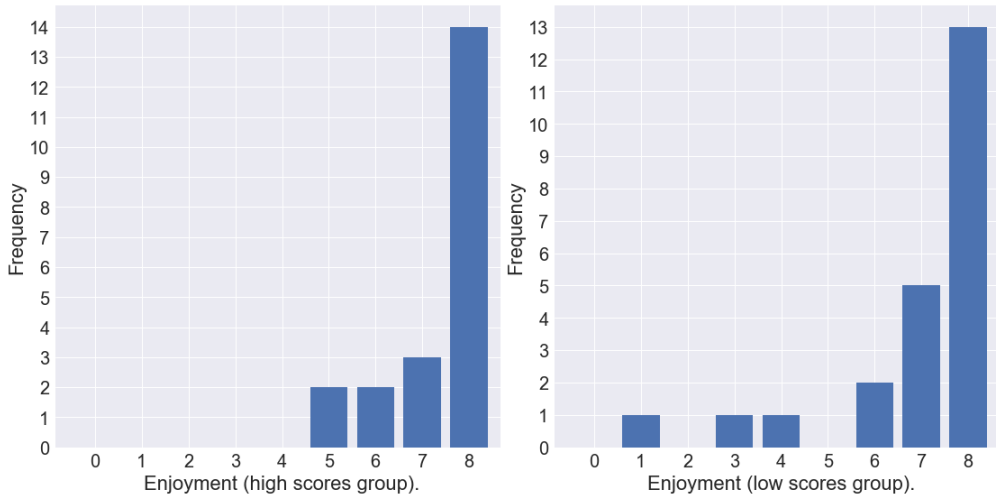

Figure 4.16: Enjoyment comparison for scored points.

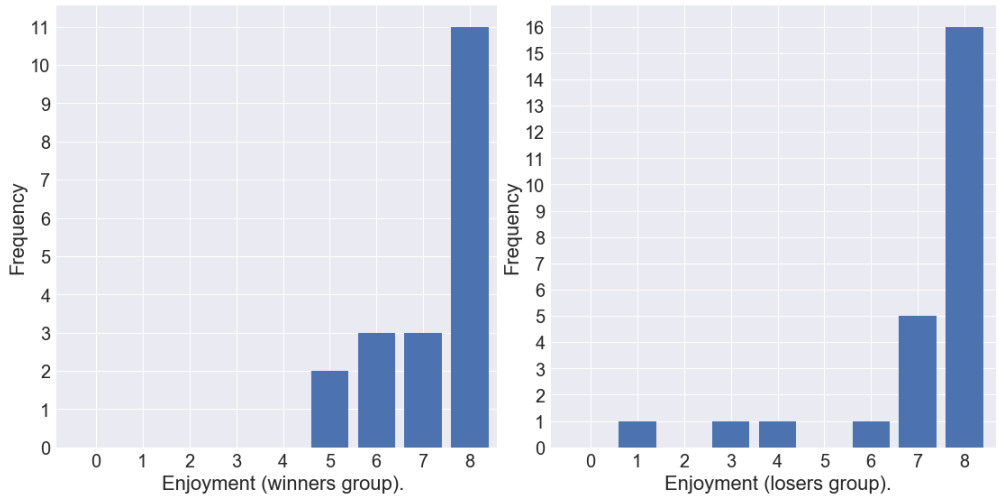

Figure 4.17: Enjoyment comparison for game outcome.

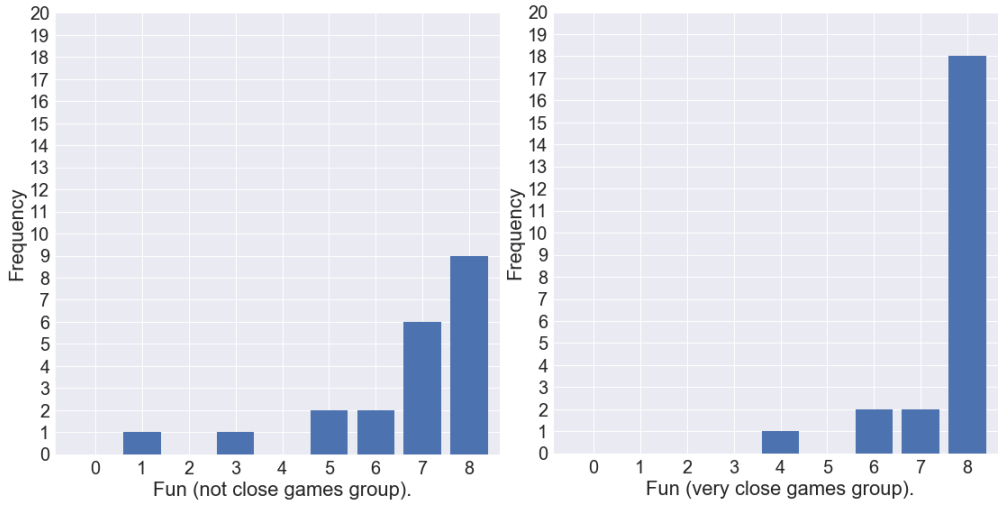

Figure 4.18: Enjoyment comparison for game outcome.

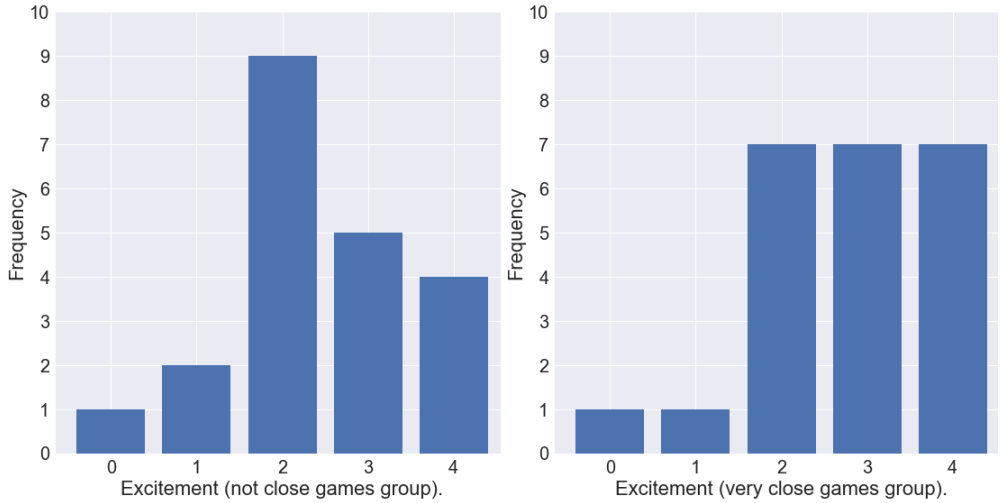

Figure 4.19: Enjoyment comparison for game outcome.

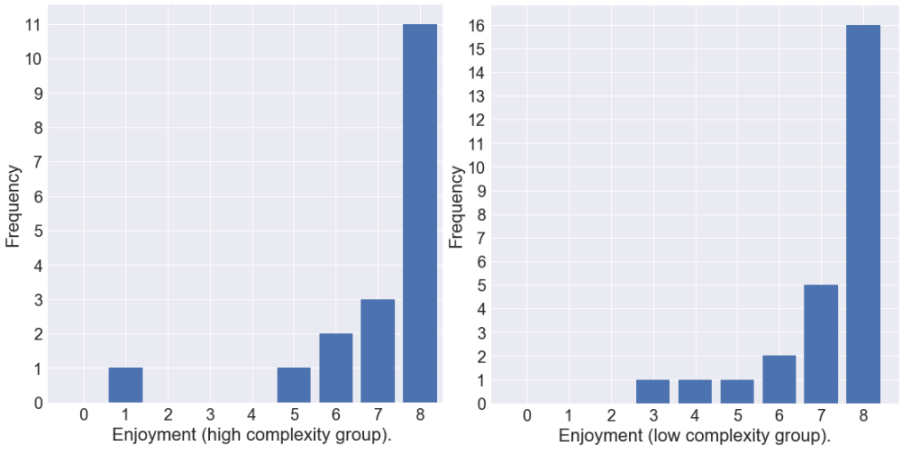

Figure 4.20: Enjoyment comparison for complexity.

4.3.9 Hypothesis: students who thought the game was very complex enjoyed the game less

The p value for the Mann Whitney U test for this hypothesis is 0.946. The data is visualized in figure 4.20.

Chapter 5

Simulation

The simulation serves three purposes: to find ways to improve the game, to demonstrate how a systematical approach (in other words, computational thinking) can be used when playing the game and to lay a foundation for future work of checking more complex properties in games like having a snowball effect (see section 7.3 on page 58).

We find ways to improve the game by programming a computer player, making it play against itself and analyzing the play patterns it demonstrates. Due to time constraints, the simulation is only done for two player games.

5.1 Methodology

The simulation and the computer player are programmed in Python. We start by creating an object that tracks the state of the game. Next we make a function that makes random moves based on who is the active player. Using this function we can play out games consisting of two rounds where two players are just making random moves. Now we can start building the computer player.

The computer player consists of three parts. Each part is used during a specific part of the game:

1. While there are still cards in the deck: the computer player does not know which cards the other player has. The computer player calculates a score for each station based on a number of factors and parameters and prioritizes coin placement cards to place coins on a relatively good station.
2. While the deck is empty and at least one player has three or more cards in hand: at this stage of the game there are too many remaining moves to compute the full state space. Instead, the computer player uses a shallow search to 'think a few moves ahead' and find a relatively good move that way.
3. While the players have at most two cards in hand: the computer player can now compute the full remaining state space. To save time, it will automatically play the best move for both players for the rest of the game so it does not compute the state space multiple times.

While building the first two parts we can still run and test the computer player by making it fall back on the random move function for the last part(s) of the game. When the computer player is finished and working correctly we optimize it with an evolutionary algorithm that makes random changes (mutations) to the computer player configuration and lets it play against copies

of itself with different configurations. Finally we let the computer play against itself with the best configuration and analyze how it plays.

5.1.1 Terminology

In this chapter we use the following terms:

- Active player: the player who makes the next move.
- Current active cycle: the cycle which would be used to count the scored if the round would end at that exact moment.
- Final active cycle: the cycle which is used to count the score at the end of the round.
- Premove: at the start of each turn, the player must place a coin on a station or remove a coin from a station. This is called the premove.

5.2 Game State

The state of the game is tracked in an object called 'State'. It has all the information about the state of the game:

- For each station, how many coins are placed there.
- For each station, who owns the coins there.
- The active player (the player who has to make the next move).
- For each switch, whether or not it is in the default position.
- Which cards are still in the deck (and can be drawn).
- Which cards the players have in hand.
- Which station is the starting station.

Additionally, it tracks some information that helps optimize the computer player. There are 11 potential cycles for the train to end up in when the score is counted at the end of each round. To optimize the computer player, some metadata about these cycles is also tracked:

- For each cycle, how many coins both players have in total on the stations in that cycle.
- For each cycle, how many switches need to be changed to make that cycle the active cycle (the one that is used when counting the score).
- For each cycle, which stations are on it.

5.3 Playing out the Game

The 'state' object has a number of functions to help play out the game. Initially, a function is called to initialize the data and deal both players a hand of cards. Then the state object is passed back and forth between two computer players who calculate a move based on the game data and metadata and call a function on the state object to make that move happen. This function also removes a card from that player's hand, draws the active player a new card if possible, changes the active player and updates the metadata. The computer players keep making moves until the deck is depleted and then the score is counted. To complete one game, this process is repeated but

with the second starting station and the other player now goes first. The winner of the game is the player with the most points over both rounds combined.

5.4 Computer Player

As mentioned in the methodology of this chapter the computer player consists of three parts corresponding to three stages of the game.

5.4.1 Stage One

In this stage, the computer player uses heuristics to pick a good move. It will always prioritize cards that place coins, as it is important to save the switch and move cards for the end of the round.

Below is a description of the algorithm for this stage. It mentions a number of multipliers and thresholds. These parameters will come into play when we use an evolutionary algorithm to find the optimal configuration for the computer player.

1. Calculate a score for each station based on the best cycle it is part of (the one with the least switches that need to be changed to make it the active cycle). The formula is $7 - SD$ (SD is the number of switches that need to be changed to make this cycle the active cycle).
2. For each other cycle each station is part of, increase the score of that station by $(7 - SD) * OM$ where OM is the overlap multiplier, which is 0.1 by default.
3. Next, the score for each station is reduced depending on how many coins it already has. The reasoning is that it is risky to put too many coins on one station. It is also important to add coins to stations that have little coins on them to prevent the other player from stealing those stations. The score is reduced by $C * FM$ where C is the number of coins on that station and FM is the fortify multiplier, which is 0.5 by default.
4. Next, the score is increased for empty stations as it is important to capture them before the other player does. The score is increased by EB for empty bonus, which is 1 by default.
5. Finally the score for each station is multiplied by SM or the station multiplier. This multiplier is 1 for each station by default, but gives the computer player an opportunity later to mutate and gain a like or dislike for specific stations.

Now that the station scores are calculated it is time to make a move. First we need to determine which station we want to place a coin on or remove one from. The computer player simply picks the station with the highest score for this, because it can always either place or remove a coin from any station. We assume that removing a coin from an opponent's high score station is just as good as placing a coin on one of our high score stations. For that individual station, the score is calculated again. Next the computer player considers the two stations that are not owned by other player with the highest scores. If the difference in score is smaller than ST (or spread threshold), the computer player will use a 'distribute 2 coins' card to target both stations if possible. If the difference is larger, the computer player will use 'place 3 coins' to target the best station, or it will use 'distribute 2 coins' and put them both on the best station. It is also possible that the computer player is still in stage one but has neither coin cards. In that case it will fall back to stage 2 algorithm (think n turns ahead) with n 0, which means it will simply make the move resulting in the highest score if the round would end right after.

5.4.2 Stage Two

In this stage the deck is empty, so the players should both know what cards the other player has. The computer player will use a recursive algorithm to 'think n moves ahead'. The algorithm starts with $n = 3$:

1. Make a list of potential moves.
2. For each move, copy the state object, make the move. This also updates the copied state to reflect it is now the other player's turn, updates the metadata, etc. If $n > 0$, recursively call the stage two computer player for each move with n reduced by 1. Otherwise, return the score of the best move.
3. If $n > 0$, evaluate all returned scores of the recursive calls. Pick the best one.
4. If $n < 3$ (it is not the initial recursive call), return the best score. Otherwise, make the move on the real state object for which the best score was returned by the recursive calls.

5.4.3 Stage Three

State space explosion occurs at around 5 remaining moves (it takes minutes to perfectly compute the best move with 4 moves remaining and hours when there are 5 moves remaining). So when there are 4 moves remaining, the computer player switches to a recursive algorithm similar to the one used in stage two, except it only stops when both players have no cards remaining and it always returns an array of moves made. This array is needed because when the recursive algorithm finishes, it simply plays out all best moves for both players. This avoids a lot of redundant computing.

5.4.4 Premove in Stage Two and Three

At the start of each turn, the player has to place a coin on a station or remove one. Fully computing every possibility for this would cause state space explosion to occur much sooner and generally slow down the computer player a lot. This is unnecessary since players often just take turns adding or removing a coin from the final active cycle. Instead, we will use a trick. We assume the computer players, once in stage 2 or 3, know exactly which cycle is going to be the final active cycle. The computer player that is going to lose will remove a coin from one of the stations on that cycle every turn, while the computer player that is going to win will add a coin to one of those stations every turn. This means we do not have to compute this because it has no bearing on the amount of points scored by the winner of that round. The exception is when there are stations with one coin on it. We use the following system of rules and a running premove bonus for each computer player (which will come into play when the score is being counted):

- The computer player can treat one station with 1 coin on it as an empty station. If it does, do not add a point to the running premove bonus. If it does not, add a point the running premove bonus for that computer player and notify the next computer player.
- If notified the computer player did not use the premove to capture a station in the last move, the computer player must ignore the best station with 1 coin on it. In other words, if capturing a specific station turns out to be the move for the computer player (by more than 1 point) and the previous computer player did not use their premove to capture a station, then retroactively determine that the previous computer player used the premove to defend one of their stations with 1 coin on it. In this case, add a coin to that station for the previous computer player and remove one point from their running premove bonus.

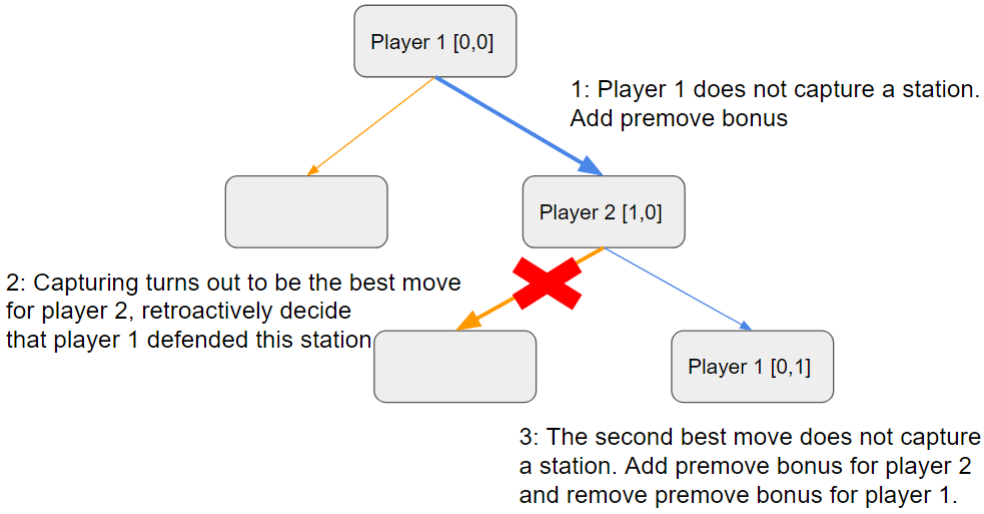

Figure 5.1: Part of a state space demonstrating how the premove is optimized by the computer player.

- Otherwise, the computer player can capture any station with 1 coin on it.

Figure 5.1 shows how this works in practise. The squares represent the state of the game. The arrows are potential moves. The orange arrows represent moves that are only possible by first capturing a station from the other player by using the premove to remove the coin from that station. The other moves are represented by blue arrows. First, player 1 does not capture a station from player 2. Because of this, the premove bonus for player 1 gets increased by 1, which means they simply used the premove to gain 1 point of advantage on the final active cycle. In the resulting game state after player 1 made their move, player 2 decides their best move is to capture a station. At this point, the premove of player 1 from the last move is retroactively changed to defend this station instead.

When the computer player is picking which move is best, each different move will have different premove bonuses for both computer players. The winner is determined by first adding the premove bonuses of both computer players to the amount of coins they scored on the final active cycle. When the score is actually counted though, the premove bonus of the computer player that lost is subtracted from the total points while the points of the computer player that won are added. This is to simulate the fact that the computer player, with perfect knowledge, knew it was going to lose, so it was removing coins from the other computer player to reduce the amount of points they would win that round. A detailed proof that this works exceeds the scope of this thesis. In any case, we think this system is at least very close in mimicking the real premove system in that it gives the computer player the option to use the premove to influence to change how many coins the players have on the final active cycle or to defend or capture stations.

5.4.5 Evolutionary Algorithm

The computer player takes a number of parameters that influence which move is picked in stage one. We use an evolutionary algorithm to find an optimal configuration for this. First we define a range of reasonable values for each parameter:

- Overlap multiplier: between 0 and 1. Default 0.1.

- Fortify multiplier: between 0 and 1. Default 0.1.
- Empty bonus: between 0 and 2. Default 1.
- Spread threshold: between 0 and 2. Default 1.
- Station multipliers: between 0.5 and 1.5.

Next, we go through a number of rounds of the evolutionary algorithm. We start with a population of 5 randomly configured computer players. For each parameter for each computer player a value is randomly selected within the above range.

1. Each computer player plays against each other computer player twice. Once where one player goes first in the first round and second in the second round and again but with the roles reversed.
2. Count the amount of wins each computer player made.
3. Eliminate the two computer players with the least wins.
4. Introduce a new randomly configured computer player.
5. Cross the best two computer players to make a new computer player and mutate it.

To cross two computer players, simply create a new one with parameters randomly rolled between the values of the parameters of the parents. A mutation of a computer player is simply a copy of that computer player with its parameters randomly changed by a random value ranging from $-x$ to $+x$, where x is the mutation strength.

This process is repeated until the exact same computer player has the most wins 4 generations in a row. Unless the algorithm takes more than 40 generations, at which point it will stop when the same computer player wins 3 generations in a row. This final computer player configuration is likely a local optimum. To discover more optimums, the entire process is repeated several times (with different mutation strengths x). Finally all the different local optimums are pooled together and play against each other to see which performs the best.

To optimize the process, the algorithm starts a new process for each game. This way CPU utilization is close to 100% on a multi core processor.

5.4.6 Results

We started by running the algorithm with the mutation strengths x 0.1, 0.15, 0.2 and 0.25, 3 times for each mutation strength. Then, all 12 computer players found this way play against each other computer player 4 times. Below, the 12 players are sorted by how many wins they scored in this last round (in case of a tie, both computer players get half a win):

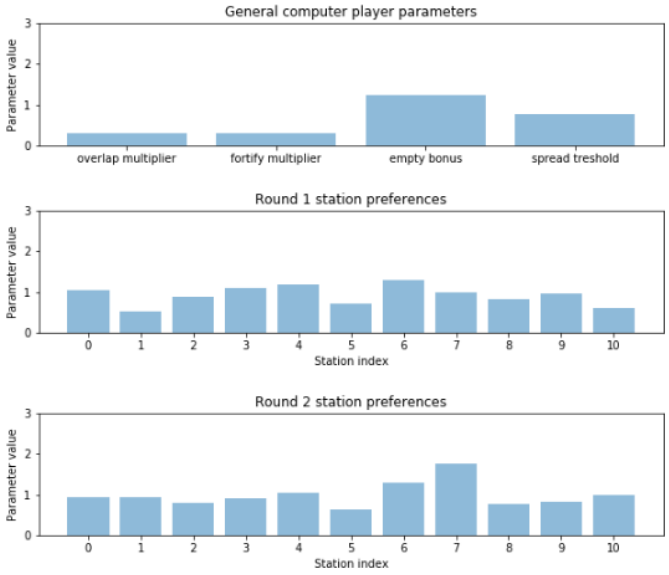

Figure 5.2: The best performing computer player configuration with 34 wins.

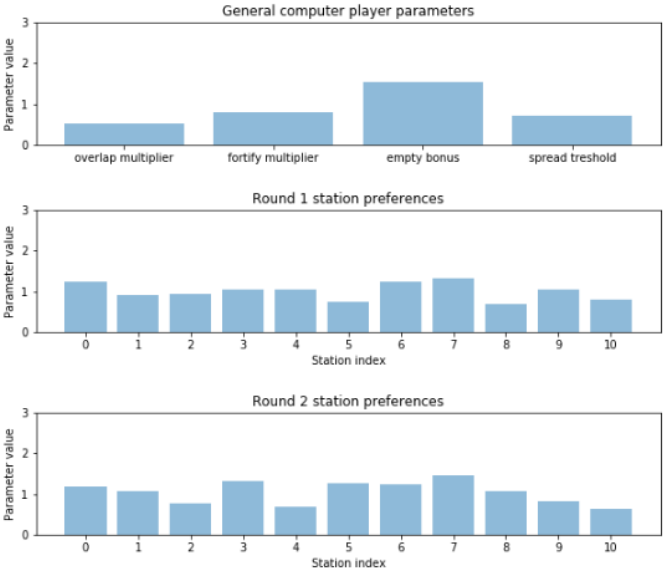

Figure 5.3: The second best computer player configuration with 31.5 wins.

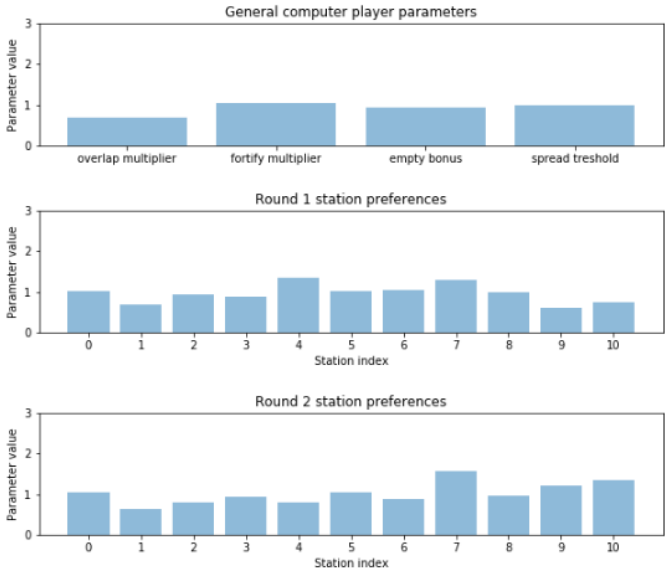

Figure 5.4: The third best computer player configuration with 29.5 wins.

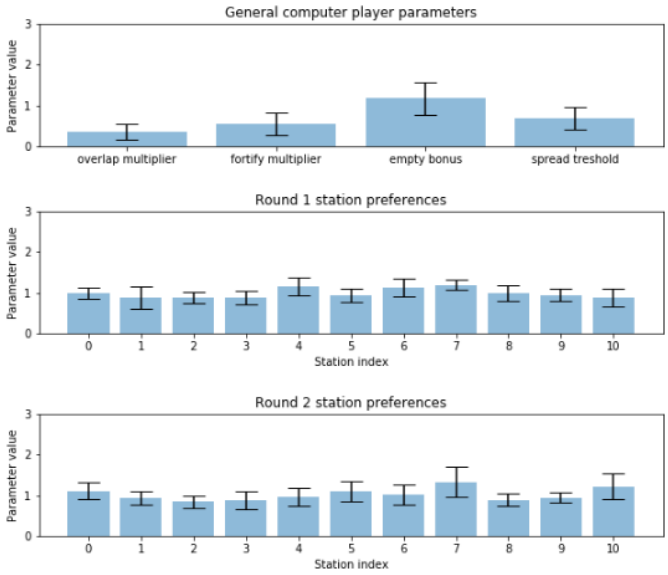

Figure 5.5: Combined graph of the top 6 computer player configurations with standard deviation shown.

The rest of the configurations can be found in the appendix. What stands out most is that the empty bonus is rather high for the top 6 configurations (see figure 5.5). This means the computer player tends to capture empty stations a little more than anticipated. Another thing that stands out is that during the second round, the computer player has a high preference for station 7 (see figure 5.6 and 5.6 on page 52).

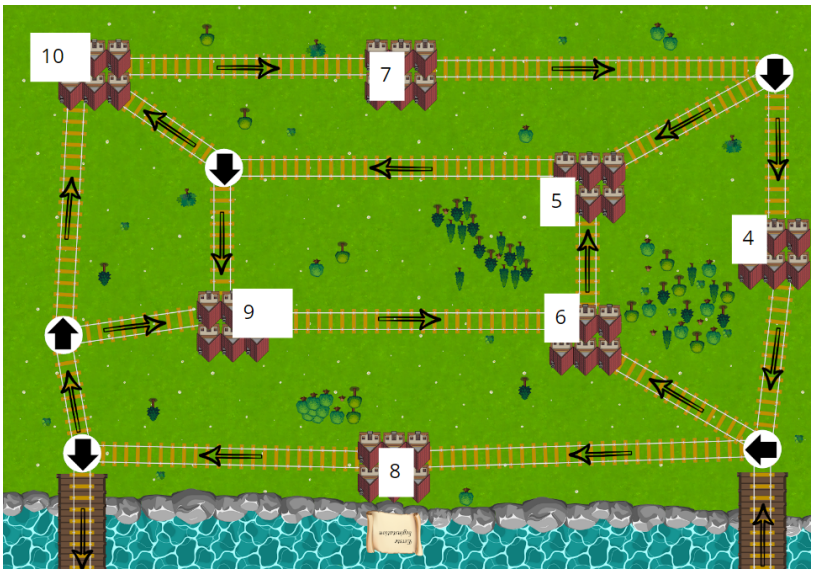

Figure 5.6: Station numbering for the computer player.

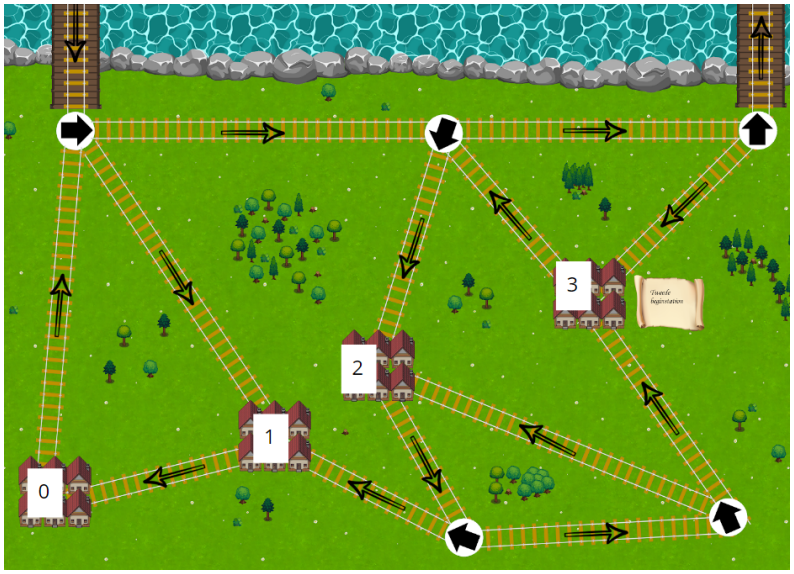

Figure 5.7: Station numbering for the computer player.

Chapter 6

Conclusions

The goal of this study is to design and create a board game (based on parity games) and to evaluate its potential to be used as an educational tool. We have designed a board game and evaluated it in several ways.

6.1 School Pilot

We tested the game with 44 students ranging from age 9 to 12 from three different schools in the outskirts of a Dutch suburban area. The students all played the game for at least 30 minutes and then answered a series of questions in a survey and a verbal interview in a group setting.

Only hypothesis 8 could be supported by the data (meaning the null hypothesis was rejected): students enjoy the game more when there is a low difference in score. This is valuable information, because it could be used to improve the game. Due to the nature of the game, many students earned 0 points. This could lead to games that do not feel very close to the students. It seems worthwhile to change the game to make it more common for the score difference to be smaller. For example, the coins from the stations leading up to the final cycle (of which all coins are awarded to a single player) could be awarded to the owner of those coins. That way, most players would usually gain at least some coins in each round and it would feel more like they still have a chance of winning even if they are behind in score.

Although the null hypothesis was not rejected for most hypotheses, it does not mean we can confirm the null hypothesis for those hypotheses either. Instead of looking at the p values, we evaluate the graphs starting on page 37 for evidence in support of the hypotheses:

- As shown by figure 4.10, some of the highest scores were only achieved by the high competitiveness group. This could indicate these students go for a more 'high roller' strategy where they place many coins on only a few stations. This is also confirmed by the variance. The low competitiveness group has a score variance of 154 while the high competitiveness group has a score variance of 268.
- Figure 4.11 suggests that students who frequently play board games said the game was less complex than the other students did.
- Figure 4.12 suggests that students who played the game in pairs said the game was a little bit less complex.

- Figure 4.13 suggests that students who played the game in pairs said the game was more exciting.
- Figure 4.14 suggests that students who played the game in groups of three said the game was more fun.
- Figure 4.15 suggests that students who played the game in groups of three had a little more strategic difficulty.
- Figure 4.16 and 4.17 suggest that students who scored a lot of points or won always enjoyed the game.
- Figure 4.19 suggests that students in games with a smaller score difference thought the game was more exciting.

In repeat studies with a larger number of students it might be possible to reject the null hypothesis for the above hypotheses.

6.1.1 Other Observations

The students enjoyed playing the game a lot. Many students expressed how much they liked the game during the pilot and asked if it was possible to buy the game later. They were happy to hear each school would get one copy of the game. This was also confirmed by the average enjoyment rating over all 44 students of 7.14 on a range from 0 to 8.

After the practise round and the correcting of misunderstandings, all students were able to understand the fundamentals of the game. Most importantly, they grasped the idea that only the final cycle counted for the scoring. Students had a good idea of which card type to use first ('distribute 2 coins' and 'place 3 coins') and which cards to save for later ('change a switch' and 'move coins').

Although the students understood the game, they still have room for growth. For example, students rarely counted the cards. By playing the game more they could learn how many cards of each type are in the deck so they can always know what cards the other player has in hand during the last few turns of each round. They also did not try to reduce the amount of points the other player would win in a round they could almost certainly not win. Sometimes they would add their own coins onto a cycle that already had many opponent's coins on it, thus only adding to their opponent's winning for that round. During the last few turns of each round, the students also seemed to have room for growth in abstract reasoning. Compared to the adult players who helped test the paper prototypes, students seemed less capable of considering all the relevant cycles and thinking ahead a few turns. This observation comes as no surprise as, according to Piaget's widely accepted theory of cognitive development [8], the oldest students in this study are just starting to develop their abstract reasoning (during the formal operational stage, starting around age 12).

6.2 Simulation

At its core, computational thinking encompasses the set of skills used when expressing a problem and a solution in a way that a computer can execute it. By building a simulation and a computer player for the game we have shown how a computer scientist might solve the problem of finding a good strategy for the game using computational thinking. The simulation and design of the computer player also reinforce that forming a good strategy for the game requires a good amount of abstract reasoning. The computer player keeps track of a lot of information which is not explicitly tracked in the physical game, such as the current active cycle, how many switches need

to be changed to make each other cycle the active cycle and which player is winning on each cycle. Keeping track of all this information and reasoning about it, preferably several turns ahead, requires abstract reasoning. According to Piaget's theory of cognitive development [8], children start developing their abstract reasoning during the formal operations stage starting at around age 12. In conclusion, the simulation as well as the preliminary study support the idea that our game can be used as educational tool to help students around age 12 and up improve their abstract reasoning, which is listed as one of the main skills for computational thinking according in most recent definitions and models (see section 2.1.1 on page 8).

Furthermore we optimized the configuration of the computer player using an evolutionary algorithm. Most of the high performing configurations have a high preference for placing coins on empty stations (more so than anticipated). This could indicate that there are not enough stations on each cycle. Especially considering that the simulation was only done for two player games and that there are even fewer stations available per player in a three player game.

6.3 Research Questions

To summarize the conclusions we reiterate and answer the research questions:

1. How suitable is the game for students from age 9 to 12?

   The students were able to quickly understand how the game is played and they enjoyed playing the game. Although students understand the game easily they still have room for growth in the area of abstract reasoning. Students show varying ability to reason about abstract concepts such as cycles, but none of them were as proficient in this as the adult players who played the game during the earlier design phases. We believe this makes the game suitable as abstract reasoning learning tool for students around age 10 and up.

2. How do factors such as age, gender, group size, game outcome, board game experience and competitiveness impact the experience of the students playing the game? The 'experience' is described in terms of how fun, challenging and complex the game is perceived.

   The main finding is that students enjoyed the game significantly more when there was a small difference in score. This can likely be explained by the fact that the game is more engaging when both players have roughly the same amount of points and it is uncertain who is going to win. When one player has many more points than the other player(s), the game is less enjoyable for both players. There was not enough data to support other hypotheses from the school pilot.

3. What strategies do the students in this study use when playing the game?

   The students had a good idea of which cards to prioritize early on in the game and which to save for later. They also thought about the up and downsides of spreading their coins out on different stations or putting them all in one spot. However, most students did not stop adding coins to a cycle where their opponent was already winning by a lot, which only increased the score of the other players. Most students were not able to think ahead one or more turns about the different cycles and possibilities.

4. How does the game relate to computational thinking?

   The main relation with computational thinking is abstract reasoning. Advanced players would quickly be able to reason about abstract concepts such as cycles, score totals on different cycles and the minimum number of switches needed to make a specific cycle the active cycle. They also need to think about how these things change in the next few turns

as both players try to score as many points as possible. We observed some of the students started to use some of these more advanced techniques.

5. How can we use the data from the pilot and the simulations to improve this game?

   The game can be improved by adding a mechanic that reduces the difference in points scored between the players. The idea is that when the game is closer, it is more engaging because the player who is behind still has a chance to win and the player who is ahead has to stay on their toes.

   Data from the simulations suggest that it is very important to capture empty stations. Perhaps this is not a problem but it is worthwhile to experiment with different boards with more stations on each cycle or perhaps a different board for three player games, where there are even fewer stations per player.

6.4 Limitations

The school pilot was done with only the primary researcher in the room with 5 to 8 students at a time. At times this was quite hectic. For most of the hypotheses we tested in the school pilot we could not reject the null hypothesis, likely due to the small amount of students that participated. In addition, all the participating schools were from the same township, so the student population in the study does not accurately reflect the population of similar age Dutch students as a whole.

Due to time constraints we only tested the simulation by writing various assert statements (to assert that the game state is correct) and running a few thousand games with a computer player that makes random moves. Ideally, the simulation as well as the computer player would be unit tested rigorously as bugs can exist and affect the outcome of the game without crashing the simulation.

Additionally, running an evolutionary algorithm on the computer player configuration was a decision that was made after the preliminary study, so this research area has not been investigated during the preliminaries of this study.

Chapter 7

Future Work

We have created a number of suggestions for future work pertaining to the following areas: game design; the school pilot and computational thinking; the simulation and the evolutionary algorithm .

7.1 Game Design

For future work in the design of this specific game we make the following suggestions:

- Design and create more modular board covers for the game that can be mixed and matched so the players can experiment with different board layouts.
- Design and create expansion cards with more interesting events. Example: last-minute switch change: this card lets you select a switch. At the end of the round when the train encounters that switch for the first time, the player who played this card may change the switch.
- Give the game a more well defined theme. It should immerse you more into a world or a story. For example, the coins could be replaced by passengers. The goal of the game would then be to pick up as many of your passengers as possible. The cards could be specific events, similar to Monopoly cards.

As we have found that students who had close games (low difference in points scored) enjoyed the game significantly more, we recommend for board games in general (as well as this specific game) to include mechanics and a scoring system that will result in more close games. This can be done explicitly in the form of a rule or mechanic that directly grants something to the player who is behind. For example, in the game we designed the player with the least points could decide who goes first in the next round. Alternatively a rule or mechanic can grant a more subtle advantage to the player who is behind. For example, all the coins the train encounters before reaching the final cycle could be awarded to the player who placed those coins. This would result in less players with 0 points scored over the course of one game and would make the player feel they are not as far behind.

7.2 School Pilot and Computational Thinking

For computational thinking and repeat studies of the school pilot with this game or another one we make the following recommendations:

- Set up a standardized and peer reviewed computational thinking test. This would be a major step forward for this research area as it would allow smaller studies to test and compare new teaching methods to other studies.
- Repeat the school pilot with a better computational thinking test, a control group, more time per student, a bigger range of ages and a wider selection of schools from all over the country. It would also be a huge benefit to get permission to record these games and later analyse how the students play: what strategies did they use and how do they improve over time.
- If it is not possible to record the game, use smaller groups of students and more researchers observing the games.

7.3 Simulation and Evolutionary Algorithm

For this area we only have two suggestions for future research. The first one is to prove that optimizing away the premove in the simulation as we did in subsection 5.1 does indeed not change the outcome of the game. The second suggestion is to expand the simulation (or another one) to investigate more complex properties. The most interesting one is the snowball effect. In a game with a snowball effect gaining a small advantage makes the game easier. So the more advantage a player has, the more easily they can gain an even bigger advantage. This makes the game frustrating for the losing player and less challenging for the winning player. This can be investigated by forcing the computer player to make mistakes at various points in the simulation. If the game has a strong snowball effect, making a mistake early on will almost always cause that computer player to lose that game, while making a mistake later on has a smaller impact on the outcome of the game.

Chapter 8

Acknowledgements

First, I would like to thank my supervisors Tom van Dijk, Karen Slotman and Mark Timmer for always providing valuable feedback, suggestions and answering my questions.

I would also like to thank the schools and the students for being open to the idea of and participating in the school pilot. In particular I would like to thank Jan Overweel from De Windroos, Willeke Berends and Jan Heijman from De Horn and Peter van den Brandhof, Loek Bakema and Barrie Hoogsteyns from De Toermalijn for taking time out of their day to help me with my research.

A special thanks goes to my parents and my girlfriend for supporting me in so many ways.

Finally, thank you to everyone who helped me playtest the game when it was just a paper prototype.

The game in its current form is for non commercial use only. Credits for the art used in the game goes to the following artists and/or websites:

- Wooden bridge by my girlfriend.
- Grass texture by vw1522191vw from Freelancer: https://www.freelancer.co.nz/contest/Toon-grass-texture-k-tileable-1437066-byentry-24194586#
- Yellow card background by unknown artist: https://hdqwalls.com/wallpaper/1680x1050/gradient-digital-art-abstract-4k
- Blue card background by Linnaea Mallette: https://www.publicdomainpictures.net/nl/view-image.php?image=227087&picture=blauwe-geweven-achtergrond
- Various plants, bushes and trees by Barbara Rivera, Casper Nilsson, Johan Charlot and Chris Phillips: https://opengameart.org/content/lpc-plant-repack
- Paper scroll (to display text) by unknown artist: https://www.pngfly.com/png-h6uzed/
- House by Fethalis: https://opengameart.org/content/pixel-house-and-fence
- Stones and rocks by microstocksec: https://www.vectorstock.com/royalty-free-vector/stones-and-rocks-cartoon-vector-8178921
- Green card background by unknown artist: http://bsnscb.com/textures-wallpapers/40160251.html
- Red card background by unknown artist: http://getwallpapers.com/collection/hd-red-wallpaper

- Water texture by Berserkerkitty on Deviantart: https://www.deviantart.com/berserkitty/art/Seamless-Cartoon-styled-Water-Texture-743787929
- Arrows (on the train tracks) by unknown artist: https://artprimshop.com/download.html
- Train tracks by me, Steven de Heus.

Bibliography

[1] Apostolellis, Panagiotis and Stewart, Michael and Frisina, Chris and Kafura, Dennis. RaBit EscAPE: a board game for computational thinking. *ACM International Conference Proceeding Series*, pages 349–352, 06 2014.

[2] Hasan Bakırcı, Arzu Kirman Bilgin, and Alper Simsek. The effects of simulation technique and worksheets on formal operational stage in science and technology lessons. *Procedia-social and behavioral sciences*, 15:1462–1469, 2011.

[3] Berland, Matthew and Lee, Victor. Collaborative Strategic Board Games as a Site for Distributed Computational Thinking. *IJGBL*, 1:65–81, 04 2011.

[4] Marina Umaschi Bers, Louise Flannery, Elizabeth R. Kazakoff, and Amanda Sullivan. Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, 72:145 – 157, 2014.

[5] Bers, M.U. The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research and Practice*, 12, 09 2010.

[6] cTwo. Denken & doen: Eindrapport van de vernieuwingscommissie wiskunde cTWO, 2015. Available at http://www.fisme.science.uu.nl/ctwo/publicaties/docs/CTWO-Eindrapport.pdf.

[7] Ghazi, Safdar Rehman and Khan, Umar Ali and Shahzada, Gulap and Ullah, Karim. Formal Operational Stage of Piaget's Cognitive Development Theory: An Implication in Learning Mathematics. *Journal of Educational Research (1027-9776)*, 17(2), 2014.

[8] Huitt, William and Hummel, John. Piaget's theory of cognitive development. *Educational psychology interactive*, 3(2):1–5, 2003.

[9] Jeannette M. Wing. Computational Thinking (presentation at Microsoft Research Asia Faculty Summit), 2012. Available at https://www.microsoft.com/en-us/research/wp-content/uploads/2012/08/Jeannette_Wing.pdf.

[10] Saul McLeod. Concrete operational stage. *Simply psychology*, 2010.

[11] Robyn Hromek and Sue Roffey. Promoting Social and Emotional Learning With Games: "It's Fun and We Learn Things". *Simulation & Gaming*, 40(5):626–644, 2009.

[12] Shute, Valerie and Masduki, Iskandaria and Donmez, Oktay. Conceptual Framework for Modeling, Assessing and Supporting Competencies within Game Environments. *Cognition and Learning*, 8:137–161, 01 2010.

[13] Shute, Valerie and Sun, Chen and Asbell-Clarke, Jodi. Demystifying computational thinking. *Educational Research Review*, 22, 09 2017.

[14] Soumela Atmatzidou and Stavros Demetriadis. Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75:661 – 670, 2016.

[15] Katerina Tsarava, Korbinian Moeller, and Manuel Ninaus. Board games for training computational thinking. In Manuel Gentile, Mario Allegra, and Heinrich Söbke, editors, *Games and Learning Alliance*, pages 90–100, Cham, 2019. Springer International Publishing.

[16] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1):127–147, Feb 2016.

[17] P. Wentworth. Can computational thinking reduce marginalization in the future Internet? In *2010 ITU-T Kaleidoscope: Beyond the Internet? - Innovations for Future Networks and Services*, pages 1–5, Dec 2010.

[18] Wikipedia contributors. Parity game — Wikipedia, The Free Encyclopedia, 2019. [Online; accessed 9-September-2019].

[19] Wing, Jeannette M. Computational Thinking. *Commun. ACM*, 49(3):33–35, March 2006.

[20] Wing, Jeannette M. Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, 2014, 2014.

[21] Yadav, Aman and Zhou, Ninger and Mayfield, Chris and Hambrusch, Susanne and Korb, John. Introducing Computational Thinking in Education Courses. pages 465–470, 04 2011.

Appendix A

Python Code and School Pilot Data

The code for the simulation, computer player and evolutionary algorithm can be found in a public repository at https://github.com/Zypp/ParityBoardGameSimulation. The data from the school pilot and Python code for analysing it can also be found here.

Appendix B

Survey

Vul in of omcirkel het antwoord:

Vraag	**Antwoord**		
Speler nummer	---		
Leeftijd	---		
Geslacht	jongen / meisje		
Groep	---		
Hoe vaak speel je bordspellen?	eens per jaar of minder	maandelijks	wekelijks
Hoe belangrijk vind je het om te winnen?	helemaal niet belangrijk	een beetje belangrijk	heel belangrijk

Geef antwoord door een van de rondjes in te vullen, bijvoorbeeld:

links ○ ● ○ ○ ○ rechts

Foutje gemaakt? Kruis het verkeerde antwoord door en vul alsnog het goede antwoord in, bijvoorbeeld:

links ○ ⊗ ● ○ ○ rechts

Hoe ingewikkeld vond je het spel?

Totaal niet ingewikkeld ○ ○ ○ ○ ○ Heel ingewikkeld

Hoe vaak vond je het moeilijk om een goede zet te bedenken?

Heel soms ○ ○ ○ ○ ○ Heel vaak

Hoe spannend was het spel?

Heel spannend ○ ○ ○ ○ ○ Heel saai

Hoe moeilijk was het om een manier te bedenken waarop je het spel kon winnen?

Heel moeilijk ○ ○ ○ ○ ○ Heel makkelijk

Hoe leuk vond je het spel?

Heel leuk ○ ○ ○ ○ ○ Niet leuk

Zou je het spel nog eens in je vrije tijd willen spelen?

Absoluut niet ○ ○ ○ ○ ○ Heel graag

Appendix C

Simulation Results (full)

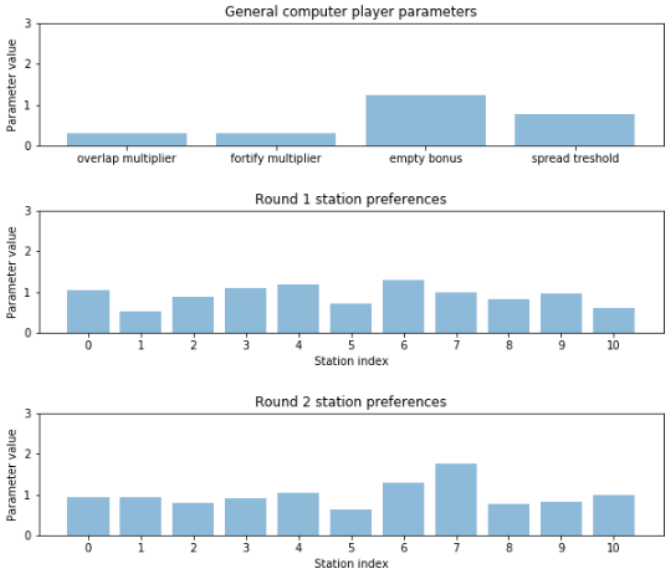

Figure C.1: The best performing computer player configuration with 34 wins.

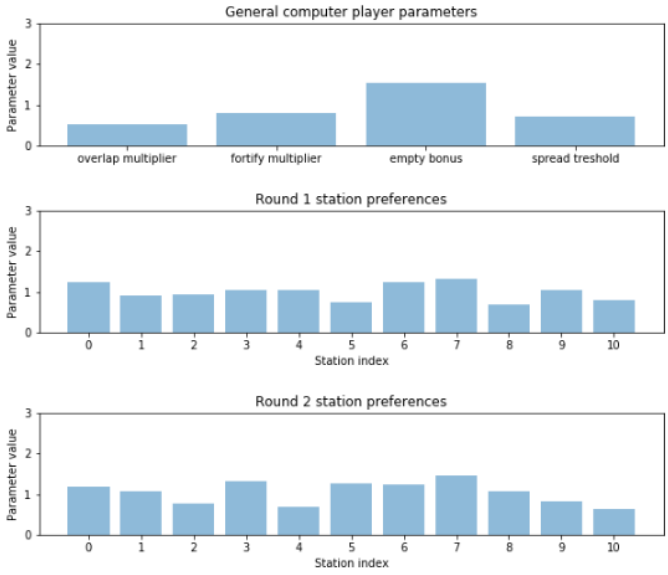

Figure C.2: The second best computer player configuration with 31.5 wins.

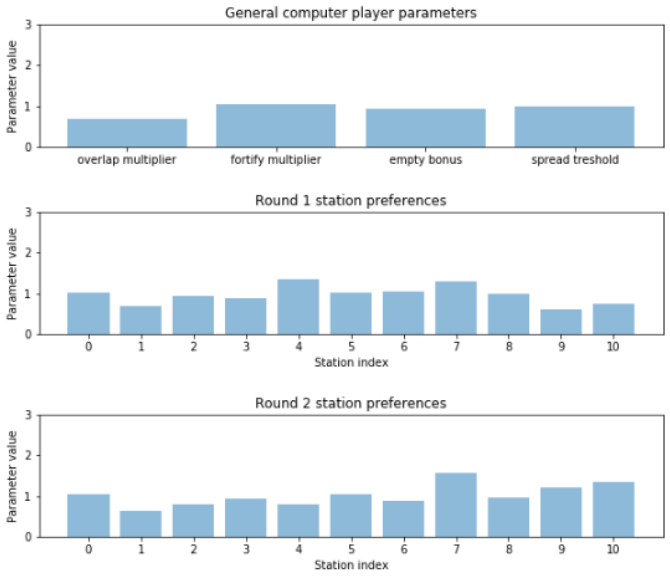

Figure C.3: The third best computer player configuration with 29.5 wins.

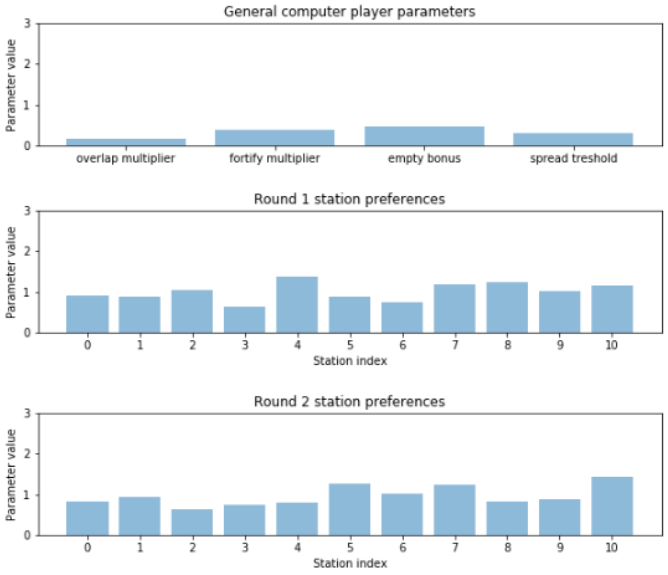

Figure C.4: The 4th best performing computer player configuration with 23.5 wins.

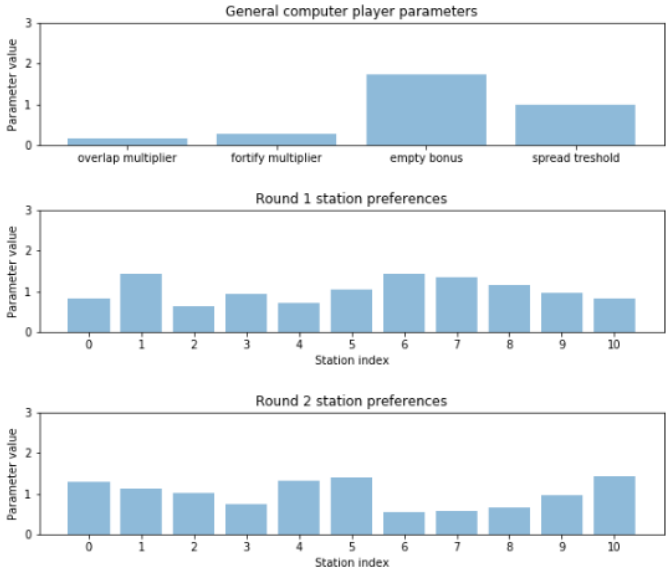

Figure C.5: The 5th best computer player configuration with 21.5 wins.

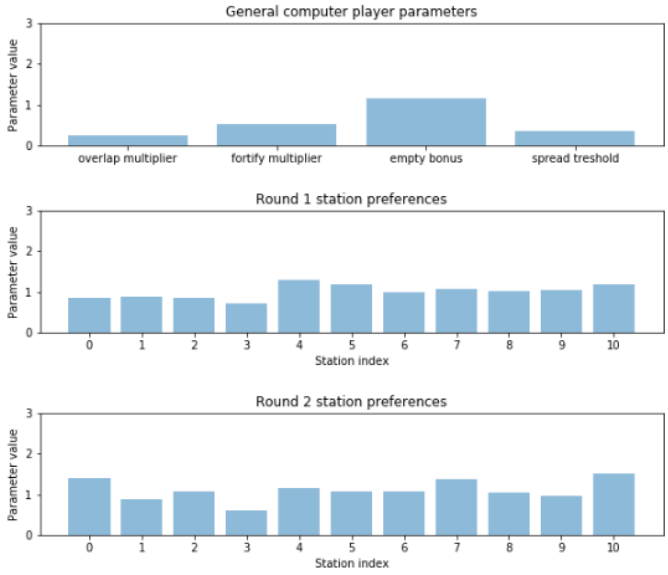

Figure C.6: The 6th best computer player configuration with 20.5 wins.

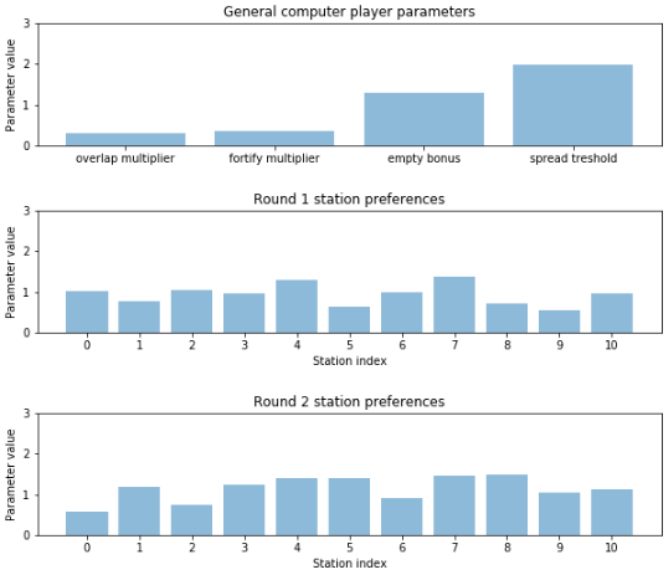

Figure C.7: The 7th best performing computer player configuration with 19 wins.

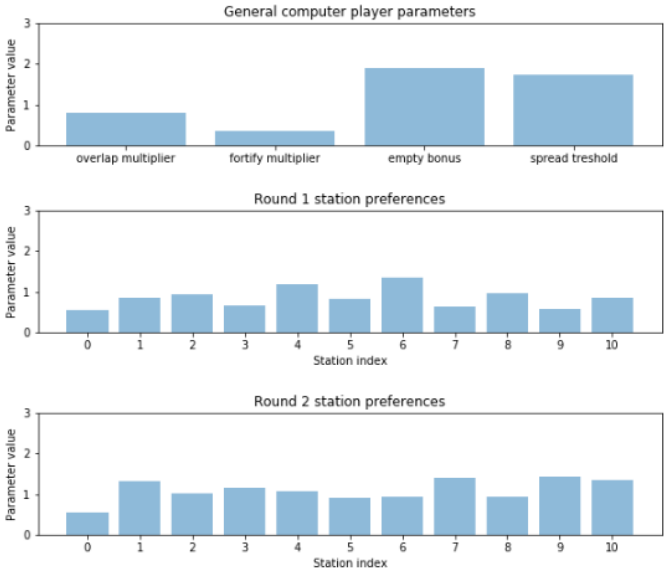

Figure C.8: The 8th best computer player configuration with 18.5 wins.

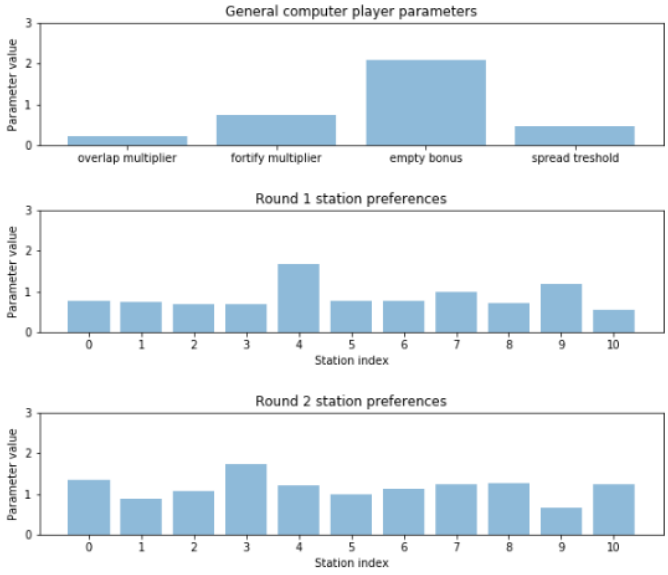

Figure C.9: The 8th best computer player configuration with 18.5.5 wins.

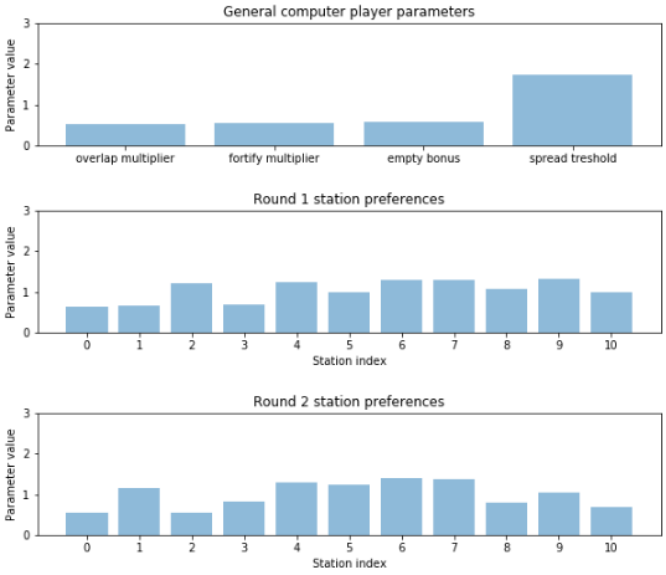

Figure C.10: The 10th best computer player configuration with 17.5 wins.

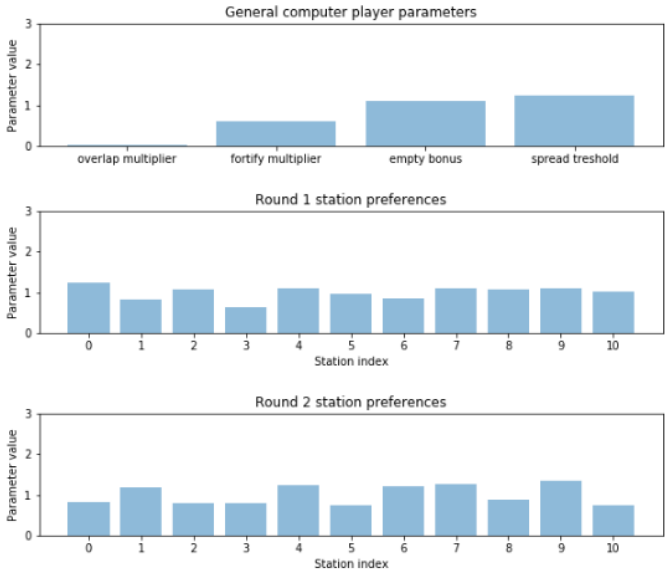

Figure C.11: The 11th best computer player configuration with 12.5 wins.

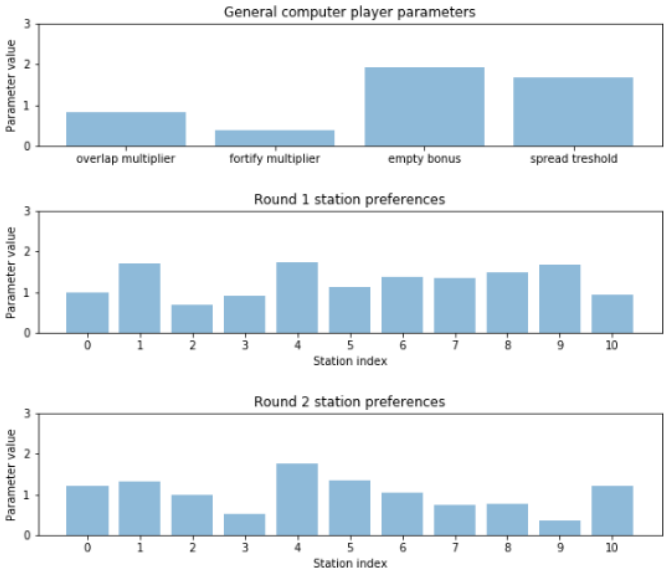

Figure C.12: The 12th best computer player configuration with 7.5 wins.