Universiteit Leiden

Master Computer Science

Analyzing AI players on different Ticket to Ride variants

Name: Mikk Õunmaa
Student ID: s2909650

Date: 30/06/2022

Specialisation: Advanced Computing and Systems

1st supervisor: Dr. Mike Preuss
2nd supervisor: Dr. Thomas Moerland

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Analyzing AI players on different Ticket to Ride variants

June 30, 2022

Abstract

Board games have gathered popularity in recent decades. The release of new and more difficult games allows players to come up with different strategies to achieve success. In this thesis the board game Ticket to Ride was analyzed. An implementation of the game was added to TAG: Tabletop Games Framework, which was later used to evaluate the performance of 4 players. One of the players used Monte-Carlo tree search for decision making and the other three players were specific to the Ticket to Ride game. Two of them could be described as rule-based players and the third evaluated every action before selecting the best.

Based on the results, one of the rule-based players, whose strategy was suggested by the community, who collected cards at the start of the game and only prioritized longer routes won most of the games. The players were also tested on different maps of Ticket to Ride game and although it reduced the effectiveness of the most successful player it was still much better than others. However, the results showed that changing the map affects the performance of different strategies as for some maps it might be better to focus on connecting the cities using the shortest routes and for others to find common parts between the cities and use them to connect the cities. Unfortunately, the used MCTS player struggled to be consistent in games and did not manage to compete regularly with the other players. Potentially, the MCTS could be improved by using different heuristic for game state evaluation and by adding tree pruning to the algorithm.

Contents

1 Introduction

In the last decades the board games have become more popular and the continuous release of new games has lead to an increase in players [1]. The released board games have more unique aspects in the game than before and this allows the players to come up with various strategies to be successful.

In this thesis one such new board game, Ticket to Ride, was implemented and added to an existing board game framework which supports the game to be played by artificial intelligence (AI) players. Ticket to Ride was selected as it has multiple strategies on how to score points and includes balancing available resources between short and long term goals. In previous works the board game had been used to test game rules with AI players and there were also many contributions using graph theory as the game map can be transformed into a graph. In addition, it had been used as a problem for the Monte Carlo tree search (MCTS) algorithm. However, the performance of the MCTS player had not been compared to other AI or human players in Ticket to Ride and this thesis tried to fill the gap by having the MCTS and three other AI players play the game in 4 player mode. The players were put against each other on various variants of Ticket to Ride to find answers to the following questions:

- What is the best performing Ticket to Ride AI player in 4 player games?
- How the performance of the players is affected when using different Ticket to Ride variants?

The rest of the thesis is divided as follows. In Section 2, the relevant background information is given, including the description of Ticket to Ride and Monte-Carlo tree search. In Section 3, the approach on how the answers to the research questions were found is described. Section 4 and Section 5 focus on the implementation of the Ticket to Ride and the used players. The conducted experiments are described in Section 6. Finally, the results of the experiments are discussed in Section 7 and the concluding words and answers to the research questions are given in Section 8.

2 Background

In this section the relevant background information is provided. The board game Ticket to Ride is introduced with its rules. In addition, a short overview of the Tabletop Games Framework and the game theory is given. The Monte-Carlo Tree Search algorithm is described together with a couple of its enhancements. Finally, an overview of the previous works related to Ticket to Ride is given.

2.1 Ticket to Ride

Ticket to Ride is a board game made by Alan R. Moon and it was published in 2004 [2]. The game can be played with 2 to 5 players and the game board shows a map of the USA. The main goal of the game is to earn more points than others and the winner is the player who has the highest score at the end. In Ticket to Ride the players are not co-operating or communicating with each other. However, due to the limited available resources and space the actions of one player can hinder the plans of other, as the player might take a card or build a route that the other player wanted. Over the years over 20 new expansions and board games have been released based on the original game, which use different areas as its board and add new aspects to the game.

2.1.1 Rules

The game board used in Ticket to Ride displays a map of North America where some of the cities from the USA and Canada are connected with routes of various colors. The routes between the cities have various lengths starting from 1 and up to 6. The game uses two types of cards: train cart cards and destination tickets. The train cart cards are referenced as train cards in the following sections. Every destination ticket has two cities from the map written on them and the number of points given to the player if the destination ticket is finished. To finish a destination ticket the player must have a connection from one city to the other using the routes on the map. In most cases, the points awarded by the destination ticket are equal to the length of the smallest possible route between the two cities. There are 110 train cards, 12 from each color (pink, white, blue, yellow, orange, black, red, green) and 14 locomotives. In the thesis the locomotives are also referenced as wild train cards. The train cards are used to obtain routes on the map. Every player represents one color and the player is given 45 train carts of that color. The colored train carts are used to show on the game board that a route is owned by the player of that color.

The set up of Ticket to Ride game is shown on Figure 1. The game area consists of a game board, 5 visible train cards, also referenced as visible train deck, discard deck, train card deck and destination tickets deck. The train card and destination ticket decks are placed face down. At the beginning of the game each player is given 4 random train cards, 45 train carts and are dealt 3 destination ticket cards of which they can choose which to keep, but they must keep at least 2. The destination tickets that are not selected by the players are added to the bottom of destination tickets deck. Each player keeps his destination tickets and train cards hidden from the other players. By the rules the starting player is the player who is the most experienced traveler and players take turns clockwise.

On each turn the player has to choose between possible actions: building a route, drawing destination tickets or drawing train cards. A player can only do one of these actions during the turn. To build a route, player must discard cards equal to the length of the obtained route. The discarded cards have to be the same color as the route and wild cards can be

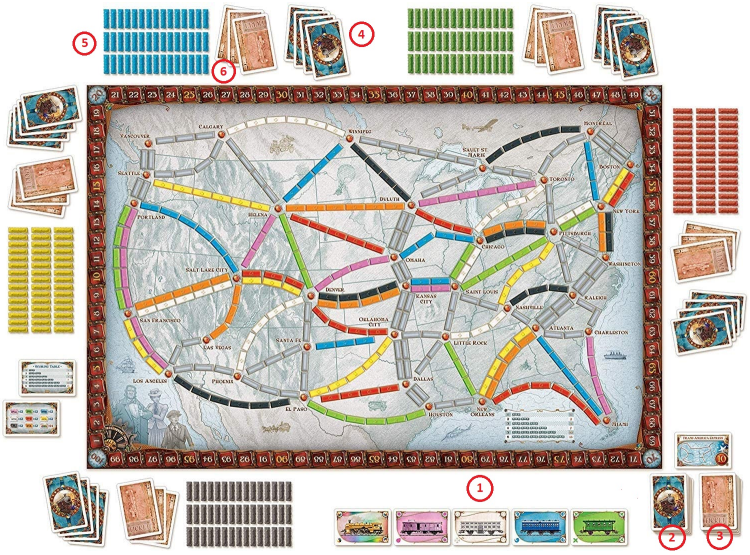

Figure 1: Ticket to Ride game setup. 1 marks the visible train cards deck. 2 and 3 represent the hidden train cards and destination tickets decks. 4, 5 and 6 are the train cards, train carts and destination tickets that are given to every player at the start.

used as a substitute in case player is missing some cards from that color. For the routes that are in grey on the map the player can decide which colored cards are used but they must be in the same color and wild cards can again be used as substitutes. In case there are multiple routes between the same cities one player can only obtain one of them. In 2 and 3 player games if one of the routes is bought the parallel routes can no longer be bought by the other players. When claiming a route, the player must place train carts, equal to the length of the route, on the route on the game board. In case the player has less train carts as the length of the route the route can not be claimed by the player.

If the player decides to draw destination tickets the player is given 3 cards from the destination tickets deck and at least one of them must be kept. The tickets that are not selected are added to the bottom of the destination tickets deck. When deciding to draw a train card the player has three options: drawing from the hidden train deck, drawing non-wild card from the visible deck or drawing wild card from the visible deck. Drawing wild card from the visible deck ends the player's turn. However, drawing from the hidden or visible deck allows player to draw additional card before ending the turn but the additionally drawn card can not be a wild card from the visible deck.

If a card is drawn from the visible deck a new card is immediately drawn from the hidden deck to replace the drawn card. In case at any time there are 3 wild cards among the visible cards the whole visible deck is discarded and 5 new cards are drawn from the hidden deck. If the hidden deck runs out of cards the discard deck is shuffled and used as the new hidden train deck.

The players get points based on the obtained routes. The scoring system is shown in Table 1. Note that the table contains scoring values also for the routes of length 7 and 8, which are not part of the base game but are used in some of the expansions. The game continues until one of the players has 2 or less train carts remaining, then every player gets one last turn, including the player who triggered the end of the game. After all players have made their last turns, they reveal their ticket cards and get bonus or minus points if they connected or failed to connect the cities shown on the ticket card. Additionally, the player who built the longest continuous route gets 10 extra points. If multiple players have a continuous route that is the longest they all get the extra 10 points. After these points are added to the scores of players the winner is the participant with the highest score. In case more than one player has the highest score then the player who completed more destination tickets wins. If there is still a tie between the players the player who has the longest continuous route wins.

Route length	Points for building the route
1	1
2	2
3	4
4	7
5	10
6	15
7	18
8	21

Table 1: Ticket to Ride scoring system.

2.1.2 Strategy

Ticket to Ride has had many competitions in the past. The game's publisher Days Of Wonder organized a World Championship competition in 2014 where over 28 000 people participated over the world [3]. The game was also part of the Board Gaming Championships events during the years of 2004 to 2019 and had couple of hundred participants every year [4].

As Ticket to Ride has various methods for scoring points it allows users to come up with different winning strategies. The differences between the strategies include when and from which deck to draw train cards, which routes and when to buy and how much ticket cards to draw and keep. As shown by Witter and Lyfold, the longest routes on the board are overvalued as they give more points per train cart and require less turns to obtain than the equivalent in shorter routes [5]. Based on this fact there was a strategy suggested by the players' community, which should lead to a win if the other players are not trying to counter it [6]. The strategy includes drawing two train cards per turn for the first 20 turns and then using the collected cards to buy the available routes that have the longest length. This allows the player to trigger the end of the game under 30 turns, which is often too quick for other players to finish all their destination tickets and

therefore they are given minus points for each unfinished ticket. The effectiveness of this strategy was reduced in the Ticket to Ride:1910 expansion made to the base game as the expansion introduced a new award for the player that had finished the most destination tickets. In addition to the new end game award, the expansion added new destination tickets and three different suggestions on which destination tickets to use in the game.

2.2 Tabletop Games Framework

The Tabletop Games Framework (TAG) is Java-based framework for creating and analysing modern board games and it is used for AI research [7]. The framework is publicly available on Github[1]. The framework consists of over 15 different board games including Uno[2], Pandemic[3] and Colt Express[4]. Compared to other frameworks that support creating AI players for board games the TAG framework focuses more on modern games. The underlying structure allows to add new games without having to re-create the whole game flow from the beginning. There are also many existing example AI players who can play the added game without additional modifications. In addition, the framework supports providing game parameters in JSON[5] file, which reduces the time and effort needed to alter the game conditions.

2.3 Game theory

If a player knows the state of the game at any stage the game can be described as a game with perfect information, otherwise it is a game with imperfect information [8]. A good example of a perfect information game is chess as both players can see at all times which pieces are still on the board. A game is classified as incomplete information game when the players lack information about other player's payoffs, resources or available strategies [9]. Ticket to Ride can be classified as imperfect and incomplete information game. It is imperfect because a player does not know which train cards are in other player's hand and therefore the exact state of the game is not known. Ticket to Ride has incomplete information because the player does not know the payoffs that the other player has by obtaining a certain route on the map.

The state-space and game-tree complexity of the Ticket to Ride for two players was analyzed by Huchler in her thesis [10]. The state-space complexity is the number of possible states that can be reached from the starting position. The Ticket to Ride state-space complexity is dependant on the number of tracks on the game board, the number of destination tickets and the number of players. The two player version of the game has much higher complexity than chess [10]. In this thesis four players were used to play

[1] https://github.com/GAIGResearch/TabletopGames
[2] https://en.wikipedia.org/wiki/Uno_(card_game)
[3] https://en.wikipedia.org/wiki/Pandemic_(board_game)
[4] https://en.wikipedia.org/wiki/Colt_Express
[5] https://www.json.org/json-en.html

the game which increases the complexity even further. The game-tree complexity is the number of leaf nodes in the solution search tree of the initial position in the game. For Ticket to Ride the exact complexity can not be calculated but an estimation can be provided. The two player game-tree complexity for Ticket to Ride calculated by Huchler was again much bigger than the complexity for chess [10]. However, the complexity is little bit smaller for 4 players due to the player having less available routes to build on average.

2.4 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) is an improvement on Flat Monte-Carlo [11]. Flat Monte-Carlo was developed for playing Go. To decide, which move to make, the algorithm selects a move and from there plays the game until the end using random actions. The reached state is evaluated and given a score. This is repeated for each action for certain amount of times and then the action which has the highest score on average is selected.

In MCTS a tree is added to the algorithm, which stores the taken actions and results. Each node in the tree represents a game state and the root of the tree is the current game state. The algorithm is based on the fact that playing a game while taking random actions once does not give a lot of information about the preferred actions, but playing the game multiple times while taking random actions will enable to infer a good strategy [12]. The algorithm is divided into four phases, which are repeated until the algorithm budget is exceeded. The budget can be either time based or based on the used resources.

The steps are following:

- Selection
- Expansion
- Simulation
- Backpropagation

In the selection step, starting from the root node a selection policy is used until a leaf node or node with unexplored children is found. The purpose of the selection is to balance selecting actions that have a good outcome (exploitation) with selecting actions that have not given good results so for (exploration). The need to balance the exploitation and exploration comes from selecting actions randomly. This can lead to an action having a bad value because an action with negative effect was selected while random actions were chosen. The exploration mitigates the risk of discarding an action which had negative effect from the taken random actions.

One of the common selection policies is the Upper Confidence Bounds for Trees (UCT). The UCT formula is shown by Equation 1 [13]. The $\bar{X}_a$ is the average value of taking action a from the node. K is the exploration constant used to balance between exploitation and exploration. N is the number of times the node has been visited and n_a is the

number of times the action has been selected from the node. The action which has the highest result from the formula is selected.

$$V(a) = \bar{X}_a + K\sqrt{\frac{log(N)}{n_a}} \tag{1}$$

In the expansion phase an unexplored action is added as a child node to the node selected in the previous step. The unexplored action can be selected randomly or by selecting actions based on the previous knowledge about them.

In the simulation a game is played from the created node in the expansion phase until the end or for longer games until certain number of actions have been taken. The actions are chosen using the play out policy. Some common play out policies are to take random actions or to use a heuristic to select from the possible actions.

After the simulation is finished the reached state is evaluated. The evaluation result is used to update the score of all the parent nodes of the created node. This is called the backpropagation step of the algorithm.

There are multiple algorithms to choose the best action after the algorithm budget was exceeded. One common choice is to choose the action node that was visited the most.

2.4.1 Enhancements

The MCTS algorithm has been one of the focuses of AI research and many enhancements have been made to the original algorithm [14]. One of the enhancements is Move-Average Sampling Technique (MAST). It was created for the general game playing competition and the idea came from the situation where MCTS would select a move which in most cases would lead to a win but there exist few good moves for the opponent which would make the selected move bad [15]. Due to the random play out, the selected move would have had a good score because it is highly unlikely that one of the good moves for the opponent was picked in the simulation phase and this forces the player to rely on the opponent to make a mistake by not selecting the best move available to them. The idea of the MAST is that for every possible action an average score is stored regardless of the state when the action was taken. The action scores are updated during the backpropagation step and the scores are used to bias towards more promising actions in the future iterations.

In large action spaces pruning the search tree can provide better results as it allows to spend more time on promising actions. The tree pruning can be classified as soft and hard [14]. In case of soft pruning the moves that were pruned in the beginning can later be still selected. In case of hard pruning the pruned moves can never be searched and selected. Progressive unpruning is an example of soft pruning created by Chaslot [16]. After a node has been visited equal to the defined threshold, only the better performing actions are kept as children. If the number of times the node is visited increases the

pruned children with the best scores are gradually added back for selection. Progressive widening is similar to the progressive unpruning, in both cases the number of available children for selection increases if the node is visited more. The difference between the two approaches is the used formula, in progressive unpruning all children are available before a node visiting threshold has reached after which only the best performing children are available. In case of progressive widening the number of available children is reduced from the first visit of the node [17].

2.5 Related works

The board game Ticket to Ride has attracted researchers with its similarity to graph problems and with the freedom for different strategies [5, 18]. Silva and others used Ticket to Ride as an example game for demonstrating AI-based playtesting of board games [18]. The authors designed 4 players with different strategies based on their own understanding of the game and based on the suggestions from the community. The players competed against each other in 2, 3 and 4 player modes and on different editions of the Ticket to Ride board game. The results showed that the success of a certain strategy depended on the used map and the number of players. In addition, during the simulation two game states were found, which were not covered by the rules. The difference with the current thesis is that the strategy of the used players differs and for some of the players used in this thesis the decision making can not be related to the decision making of the human player. In addition, the thesis only focuses on the 4 player version of the game.

Witter and Lyford analyzed the game board of the Ticket to Ride from the perspective of graphs [5]. The cities and routes between cities were perceived as nodes and edges between the nodes in a graph. The analysis of the game board was only done for the Ticket to Ride USA map. The authors used the same 4 agents designed by Silva in his research [18]. The results showed that the longer the route gets the more profitable it is to the player. A new scoring system was proposed, where the points awarded for a route are proportional to the size of the route. In addition, the simulated games were used to find out the most common destination tickets leading to a win and the difficulty of completing a certain destination ticket. Since the players in this thesis are used on different maps the findings on destination tickets and routes can not be used in the implementation of the players.

In her Master's thesis Huchler focused on different variations of MCTS players and their success in Ticket to Ride [10]. The results showed that adding progressive unpruning and progressive bias to the MCTS algorithm lead to better performance from the player. However, all experiments were conducted using only various variations of MCTS enhancements and there was no indication on how good the best MCTS player was compared to human or strategic AI player. In their work, Dinjian and Nguyen devise a guideline on how to teach AI to play Ticket to Ride [19]. They suggested using curriculum learning for the AI, as the environment is too big to learn directly. Unfortunately, the authors were unable to demonstrate the effectiveness of the taken approach because of poor design

and implementation decisions.

3 Approach

To find the AI player that achieves good results in Ticket to Ride an implementation of the game and the players is needed. The Ticket to Ride implementation was done in Java as an addition to the TAG framework. Although the TAG framework already had several implementations of players, 3 new players were added to the framework which were specific to the Ticket to Ride game. Two of the added players were rule-based and one evaluates all actions to choose the one with the best result. The logic for one of the rule-based players was taken from online forum, where the strategy to win games was described. The logic of the other rule-based and evaluation players was taken from online from other projects that had implemented the Ticket to Ride game. For the fourth player the existing implementation of MCTS was used. Before the MCTS player was put up against the other players an optimization of the algorithm parameters and heuristic functions was needed to find the best version of the player. The Sections 4 and 5 go into more detail about the implemented game and the used players.

The effectiveness of the players in Ticket to Ride is analyzed by having the 4 players compete against each other in 4 player games. In addition to the base game map, the maps and destination tickets from the Europe and Germany edition of the Ticket to Ride were used to gather data from other maps as well. The additions added by the Ticket to Ride: 1910 expansion to the base game were also included in the experiments. The players can play any given map without modifications as long as the set of possible actions stays the same. The Europe and Germany editions have additional rules when obtaining routes but they were not included in the implementation to keep the action set the same for all maps. The division of destination tickets into two categories was also excluded for both of these maps. The results of the players on the three maps and with different destination ticket sets were compared to find the player which achieved the best results overall. In addition, the effect of the maps and various destination tickets contents on the players' performance was analyzed.

3.1 Ticket to Ride variants

The distribution of routes and the destination tickets of the base game of Ticket to Ride is shown on Figure 2. In case of parallel routes only one of them was considered on the graph as the player can buy one of them and the parallel routes have equal length. By far the routes with length two are the most common. There are 9 routes with length 6 which means that in best case scenario the player can use all the train carts in 8 turns to buy 7 routes with length 6 and one route with length 3. The destination tickets follow a normal distribution meaning that the most common tickets are the ones that award 9-11 points and the ones that give very few or a lot of points are quite rare. The median value

of the destination tickets is 11.

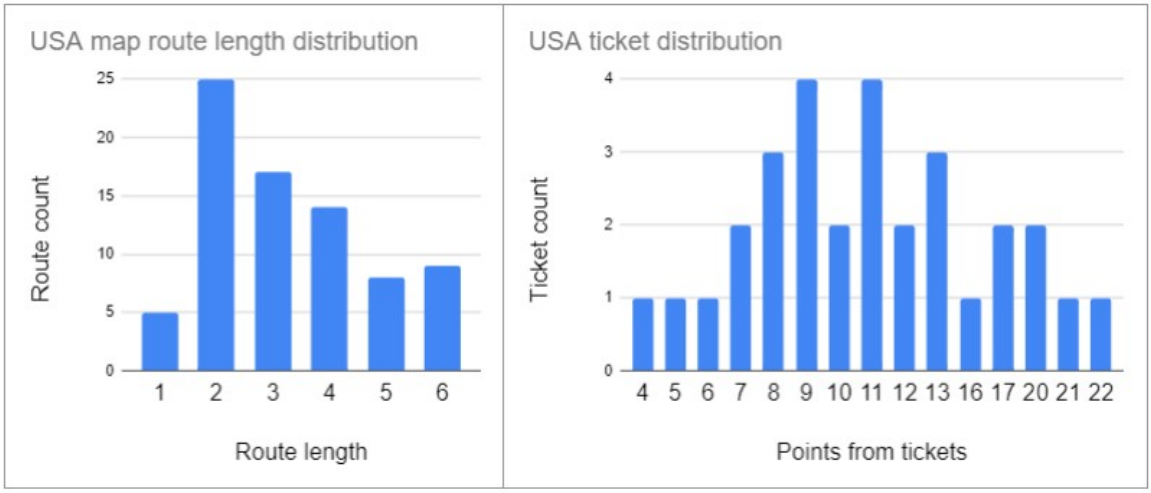

Figure 2: USA map route and destination tickets distribution. The tickets distribution on the right follows the normal distribution. On the left, the most common routes are with length 2 but there are considerable amount of routes with length 6.

The Ticket to Ride: Europe map is shown in Appendix A. The distribution of routes and destination tickets of the map of Europe is shown on Figure 3. Compared to the USA map it has only 3 routes with length 6 or more whereas the USA had 9 such routes. This also means that in Europe it takes at least 9 turns to spend all train carts and trigger the end of the game and it requires that none of the longer routes have been claimed by the other players. The routes with length two are again the most common but there are also a lot of routes with length 3 and 4. The destination tickets for the Europe map were taken from online [20]. In the Europe version there are long and short tickets. One long ticket is given to every player at the start of the game among the other given tickets and then the player can decide to keep the ticket or not [21]. During the game the long tickets can not be drawn. However, in the implementation added to TAG only one destination ticket deck is supported and therefore the two types of tickets are merged into one deck and tickets are dealt randomly from the merged deck at the beginning of the game. Compared to the USA, the tickets with length 8 are most common and the tickets with other amount of points are distributed quite evenly. The median value of the tickets is 8, which suggests that on average compared to the base game less points are received from the destination tickets.

The Ticket to Ride Germany map is shown in Appendix B. Germany has 6 cities on the map that act as dead-ends, meaning that these locations can only be the end or start of a destination ticket path but they can not be in the middle of connecting two cities. Similarly to the base game, the Germany map also has many double routes, it also has some triple routes. The same rules are applied to the triple routes as to the double routes. The inclusion of double and triple routes possibly reduces the contention for some

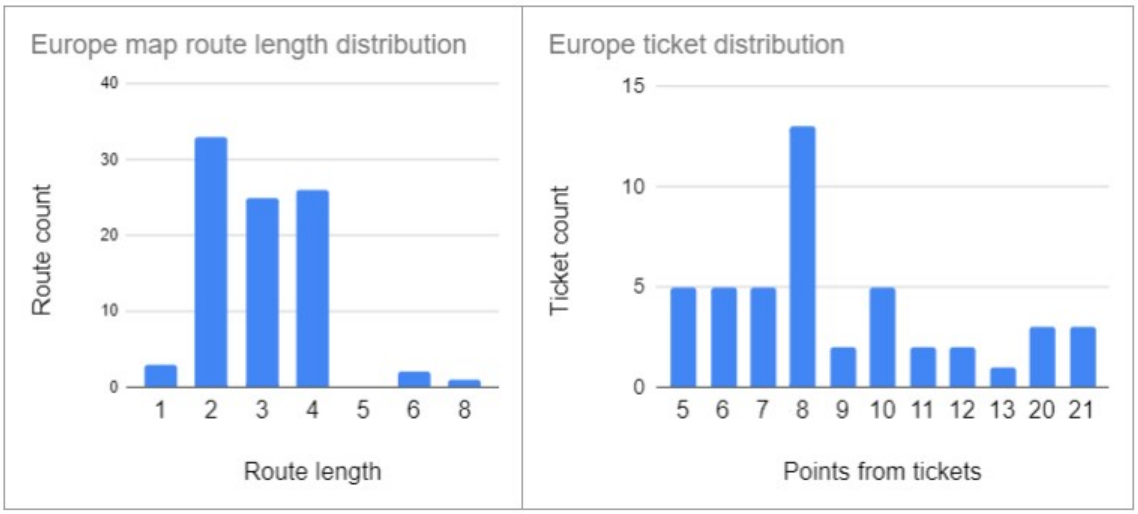

Figure 3: Europe map route and destination tickets distribution. The tickets that award 8 points are by far the most common. Compared to the USA, there are very few long routes and routes with length 2, 3 and 4 are very common.

of the routes on the map. The distribution of routes and destination tickets of the map of Germany is shown on Figure 4. Compared to the previous maps the most common routes are with length 3 and there are a lot more routes with length 1. Although, there are a lot of routes with length 1 it is possible to use all train carts only in 7 turns by claiming 4 routes with length 7, 2 routes with length 6 and 1 route with length 5. The destination tickets for the Germany were taken from online [22, 23]. The destination tickets in Germany are also divided into long and short tickets like in Europe version but the long tickets in Germany can be drawn during the game [24]. However, the tickets were kept as one deck in the experiments for the same reason as the Europe tickets. Compared to the other maps, the Germany edition has more tickets in general. Furthermore, the tickets in Germany are distributed over longer point range from 2 to 22 and there are 28 total tickets which give 7 points and less, which means there are many short tickets that can be finished quickly.

The Ticket to Ride base game (map of USA) has received an expansion called the Ticket to Ride: 1910. The destination tickets added by the expansion were taken from BoardGameGeek [25]. Besides the new destination tickets the expansion also contains 4 changes to the points of the existing destination tickets. The point modifications to the destination tickets were only used when the other added destination tickets were used. In the rules of the expansion three ways (1910, Big Cities, Mega) are proposed on how to compose the destination tickets deck and whether to award the player with the longest route or the player with most finished destination tickets or both with extra points at the end of the game. The proposed three destination ticket sets were used as three separate experiments. In case of 1910 and Big Cities the player with most completed

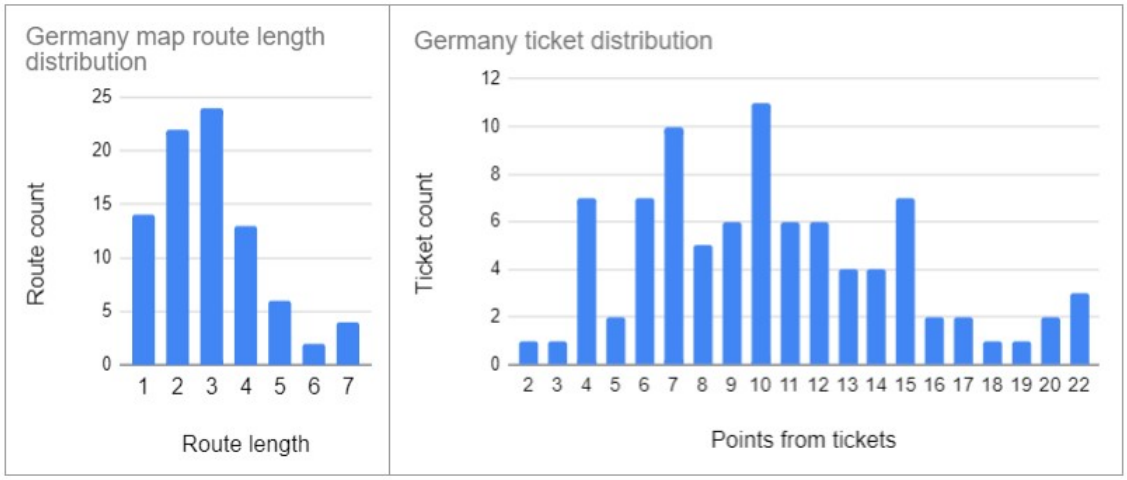

Figure 4: Germany map route and destination tickets distribution. Similarly to the USA, in general the tickets follow the normal distribution but there are some points which have more tickets. The routes with length 2 and 3 are most popular but compared to the other two maps there are a lot more routes with length 1.

destination tickets got 15 bonus points at the end of the game instead of the player who had the longest continuous route. The destination tickets distribution of the 1910 set and Big Cities set are given on the Figure 5. The 1910 destination ticket deck has more shorter routes than the original but the distribution still follows normal distribution. The Big Cities destination tickets are quite evenly distributed from 5 points to 12 points compared to the original deck which followed normal distribution. In addition, in Big Cities the shortest tickets start from 5 whereas in the original and 1910 there are tickets that give 2 or 4 points. The number of long destination tickets that give more than 17 points stayed roughly the same for the original and 1910 and Big Cities variants.

The final variant to compose the destination tickets deck was called the Mega game. It has all the tickets from the Ticket to Ride: 1910 expansion and the original game. At the end of the Mega game the players with the longest continuous route and the most completed destination tickets are given bonus points. The distribution of the tickets is displayed on Figure 6. Compared to the original the Mega game has 39 more destination tickets. The most common tickets are the ones that give 7 to 11 points whereas in original game the most common were tickets with 9 to 11 points. Compared to the original game there are also more tickets that award 17 or more points but the chance to get such ticket is a little bit smaller.

4 Game implementation

Classes that represent the game state, forward model and actions have to be implemented to add a new game to the TAG framework [26]. In addition, the game specific

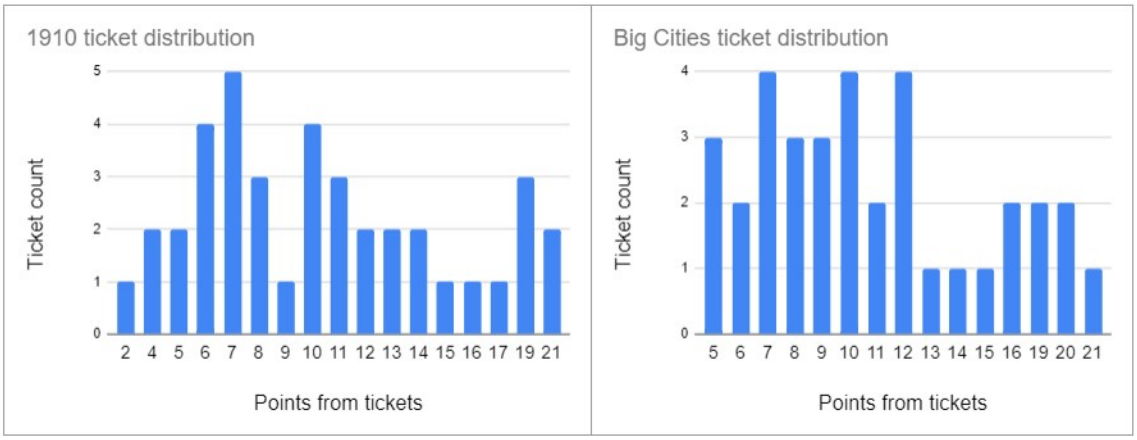

Figure 5: 1910 and big cities destination tickets distribution. In case of the 1910 tickets, the tickets follow similar distribution but there are more shorter tickets. In case of Big Cities, the shortest tickets are longer and the distribution from 5 to 12 points is quite even.

data and parameters are required. To help with the implementation there are already abstract versions of the needed classes that can be extended. The general overview of the components and the relation between them is shown on Figure 7.

4.1 Game data

The framework supports providing game data in the form of JSON files. The files are parsed automatically and the file contents are read by the forward model when initializing the game. However, to use the JSON parser created by the framework the file has to have a certain structure. Unfortunately, the route connections between the cities was not possible to define using the existing JSON structure. Therefore, a Ticket to Ride specific JSON parser was created based on the existing parser. Figure 8 shows a part of the JSON file which holds the Ticket to Ride game board data. On the figure is the city Vancouver with one of its routes to Calgary. The previously existing parser was unable to read the data from the *neighbours* JSON property because it expects all property values as pairs of property type and property value and only simple property types are supported like *String*, *Integer* and *Vector2D*. Since such representation of neighbours is specific to Ticket to Ride the parsing of the *neighbours* values was not added to the general parser.

Two types of data was stored in the JSON files. The first file contained the layout of the used map. The other file contained the description of the destination tickets. Other inputs of the game were hard-coded, although it would also have been possible to read the train cards, number of trains and scoring system from the files. Hard-coded approach was preferred since these inputs were very likely to stay the same during all the experiments and moving them to the files would have been unnecessary time cost.

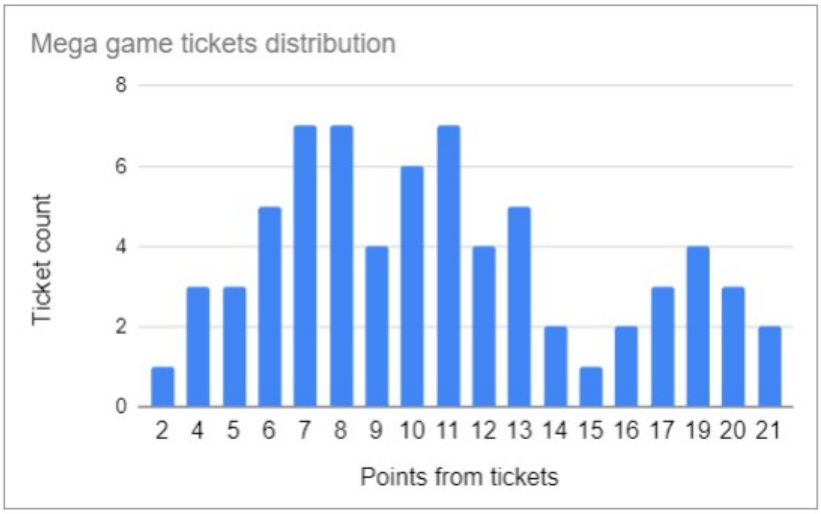

Figure 6: Mega game destination tickets distribution. Compared to other maps there are a lot more tickets but they are distributed quite similarly as in the base game. Although there are more longer tickets, the addition of other tickets reduces the probability to draw a ticket that gives 17 or more points.

4.2 Game state

The state of the game is stored in the game state object. For Ticket to Ride the game state holds the information about the current scores of the players and all the player areas and a general area. The current score of every player is available to each player. However, during the game the score is equal to the number of points awarded by claimed routes. The points from destination tickets and points for the longest continuous route are only added at the end of the game to the scores.

General area consists of the game board, destination tickets deck, discard deck, visible and hidden train cards deck. Game board keeps track of the routes between the cities and the owners of the routes. The contents of the destination tickets and hidden train cards decks are not visible to players. In the rules it is not mentioned if the discard deck is visible or not to the players, since it can be tracked what cards other players have discarded and what cards are discarded from the visible train cards deck it was decided that the discard deck is visible to the players.

The player area contains the information that is specific for every player. It holds the cards that are in player's hand, the selected destination tickets of the player and the counter for keeping track of how many train carts the player has left. In addition, if player has decided to draw destination tickets there is an additional temporary destination tickets deck for holding the drawn tickets until player has selected which tickets are kept and added to the player's destination tickets. All contents except the train carts counter are only visible to the player itself. The reason for allowing other players to look at the number of used

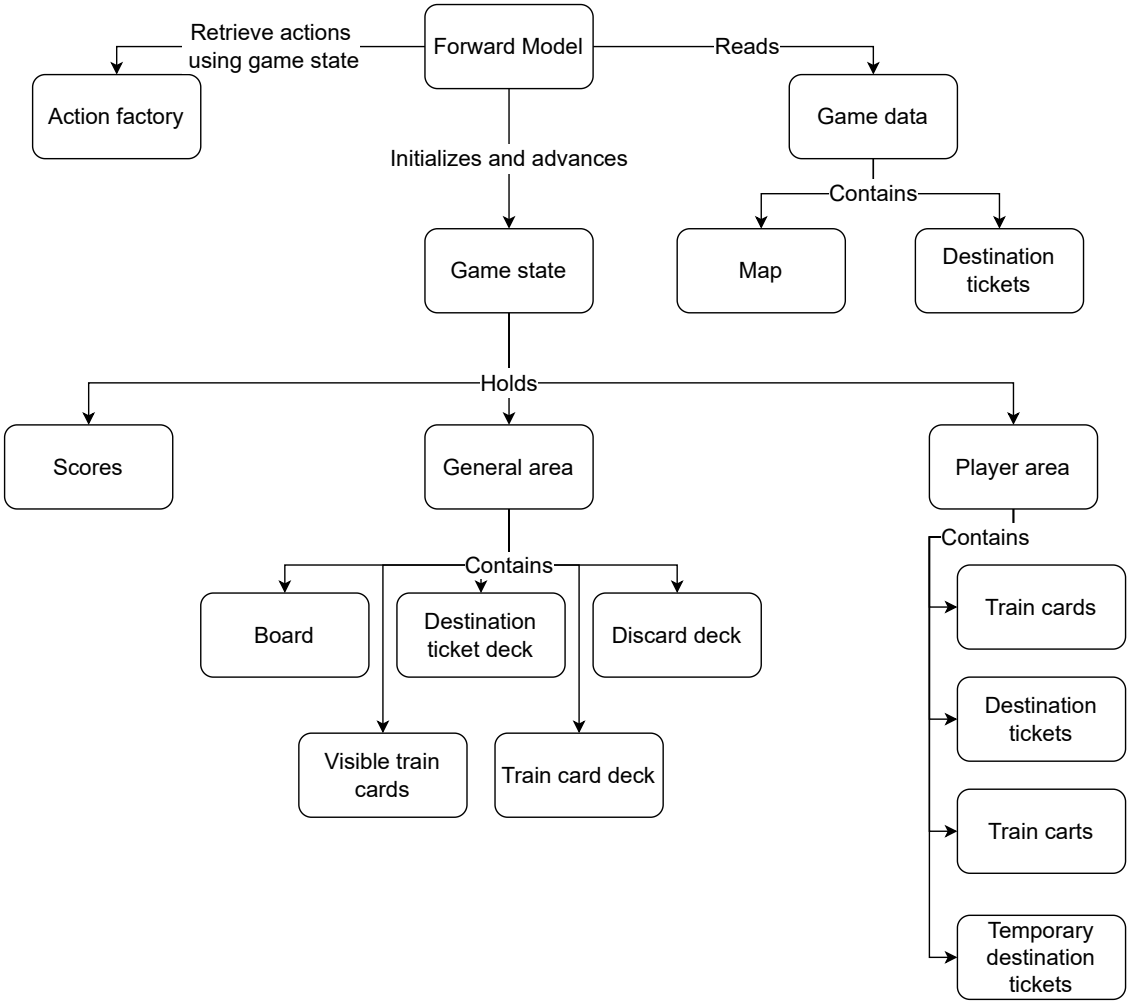

Figure 7: Overview of the game implementation components.

train carts for each player is that the number of train carts left can also be calculated by looking at the current state of the game board and therefore it is not reasonable to hide the number from others since they can deduct the value anyway.

During testing of the Ticket to Ride implementation some states were encountered which were not covered by the rules and stopped the game from continuing. The same erroneous states were also encountered by Silva [18]. The solution for when there is not any cards that can be drawn and any routes to claim was to create a new action which allowed player to do nothing in its turn. The other problematic state was when the visible train deck always had at least three wild cards due to the lack of other cards in hidden train deck and discard pile was empty. By the rules the visible train cards had to be discarded and new cards drawn to replace them but since there was not enough non-wild cards it lead to endless loop of discarding and re-drawing cards. To solve this the discarding of the visible deck was disabled in case there was not at least 3 non-wild cards in the hidden

Figure 8: Excerpt from the Ticket to Ride JSON file with board data. The figure displays how a route from Vancouver to Calgary with length 3 and with grey color (cards with any one color can be used to claim it) is defined.

deck.

4.3 Forward model

The forward model has two main tasks. Firstly, it is responsible for setting up the game and creating the first game state. It reads the data from the JSON files and creates the board and destination tickets deck based on the contents. Then the initial 4 train cards are given to the players. The starting player is the player who is in the first player position, which is different from the rules, where the starting player was supposed to be the one with the most experience as the traveller. To change the order of players the players must be put in different starting positions. In the board game version of Ticket to Ride the destination tickets are also handed out in the set up phase. However, the framework does not support asking for player input while initializing the first game state. Therefore, the first turn of the game is used to allow players to select their destination tickets. This is done by providing the players only with the draw destination tickets action. Starting from turn 2 all types of actions are possible again.

The second task of the forward model is to provide player with the possible actions and apply the player chosen action to the game state. After the action is executed, additional checks are performed to find out if further changes to the game state are needed. In case 5 or less cards are remaining in the train deck, the discard pile is shuffled and the cards are added to the bottom of hidden train deck. If the visible train cards deck has less than 5 cards then the missing cards are added to it. Also, the visible train cards deck is discarded and new cards are drawn to replace them if there were at least 3 wild train cards in the visible train deck. Finally, the train carts of the players are checked to see if there is a player who has 2 or less train carts left and the final round should be started.

4.4 Actions

The actions that the players can take are provided by the action factory. There are two types of actions, one that takes the whole turn and other that requires the player to take subsequent action. All the actions have direct effect only on the player who took the action. In total there are 10 possible actions and the relations between the actions are shown on Figure 9.

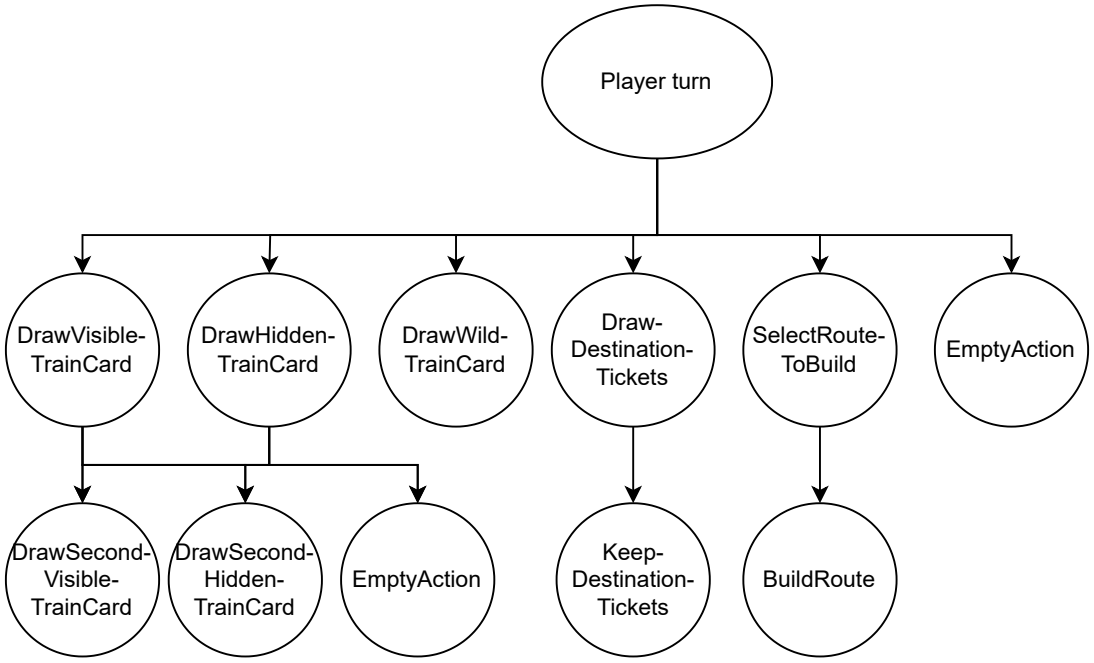

Figure 9: Relations of the actions the player can take. First row is the actions that are available at the start of the turn and second row actions are available if a certain action from the first row was taken.

The actions in the first row are the actions that can be provided to the player at the beginning of the player's turn. The arrows are used to indicate which actions are possible after a certain first action has been taken. At most a player has to choose two actions

that are taken during the turn. *EmptyAction* is only used when no other action is possible to enable players to continue playing the game. One of the situations where it is used is when all players decide to collect cards and it leads to hidden and visible train card decks to run out of cards after a player has drawn a card as the first action of the turn.

4.4.1 Claiming a route

Claiming a route is divided into two actions: *SelectRouteToBuild* and *BuildRoute*. The decision to divide it into two parts was made after the first games were played with the MCTS player. With a single action for claiming a route the player often had over 100 actions to choose from and with the budget restriction it was unable to simulate enough iterations to identify the best action and the player ended up with choosing an action randomly.

SelectRouteToBuild is the first step of claiming a route on the board. This action is only available if the player has enough cards to obtain any of the available routes on the map. For every route that can be obtained by the player there is a separate action. The action contains the route that is going to be claimed and the player claiming the route. After the action is taken the player is assigned as the owner of the route and the number of remaining train carts is reduced by the length of the route. The reason for changing the owner of the route when this action is executed was that otherwise there would be no change in the game state after this action and the heuristic function evaluation of the state would not change.

After *SelectRouteToBuild* action has been taken the player is provided with at least one *BuildRoute* action. Every *BuildRoute* contains the route that was obtained in the previous action and a list of train cards to discard. The purpose of *BuildRoute* is that the player can decide which cards were used to obtain the route and favor train cards of certain color when claiming a route without a certain color. For every possible train cards combination that can be used to build the route a separate action is provided. When *BuildRoute* is executed the cards that were included in the action are taken from the player's hand and added to the discard deck.

4.4.2 Drawing a train card

In total, 5 actions were created to draw a train card. The difference between the actions is in which deck is used to draw a card and whether it is the first or second action in the turn. After the draw a card action is executed the card is drawn and added to the player hand. In case there are multiple cards with the same color in the visible train cards deck then still multiple actions are created for drawing the same color but the index at which the card is drawn from the deck is different.

If it is the beginning of the player's turn then the player is given the possibility to draw a card, among claiming route and drawing destination tickets actions, by choosing from *DrawHiddenTrainCard*, *DrawVisibleTrainCard* and *DrawWildTrainCard* actions, provided

that the action conditions are met. *DrawHiddenTrainCard* is possible when there is at least one card left in the hidden train cards deck. *DrawVisibleTrainCard* is provided when there is at least one card in the visible train cards deck and *DrawWildTrainCard* is provided when there is a wild card in the visible train cards deck. *DrawHiddenTrainCard* draws the top card from the hidden train cards deck. *DrawVisibleTrainCard* action contains the color and index of the card in the visible train cards deck. The index is used to draw the card from the deck at the given location. The train cards with wild color cannot be drawn using the *DrawVisibleTrainCard* action. *DrawVisible-* and *DrawHiddenTrainCard* actions do not end the player turn, since a player can draw two cards in the turn if the first card is not a wild card from the visible train cards deck. *DrawWildTrainCard* is used to draw a wild card from the visible train deck and it ends the player's turn and it is only available as the first action of the turn.

The *DrawSecondHiddenTrainCard* or *DrawSecondVisibleTrainCard* are provided to the player, provided that there still exists cards in the hidden and visible train cards decks, if the *DrawHiddenTrainCard* or *DrawVisibleTrainCard* action was taken in the turn. *DrawSecondHiddenTrainCard* and *DrawSecondVisibleTrainCard* are used to draw a second card in the turn and they act and contain the same information as their counterparts *DrawHiddenTrainCard* and *DrawVisibleTrainCard*. The player does not have to draw both cards from the same deck, it is possible that the player drew the first card from the hidden deck and second from visible or vice versa. In case both the hidden and visible deck are empty after the player drew the first card the player is provided with *EmptyAction* that ends the turn.

4.4.3 Drawing destination tickets

DrawDestinationTickets draws three destination tickets and gives them to the player. At the start of the game this is the only action for all players in the first turn. This action also does not end the player's turn since the player has to decide which destination tickets to keep. The player is given a choice between multiple *KeepDestinationTickets* actions. Each action offers the player different combinations from the three given destination tickets. The tickets that the player decided to keep are added to the player destination tickets deck and other tickets are added to the bottom of the destination tickets deck.

5 Players

In total, 4 players were used to play the game. These were MCTS based player, one player whose strategy was taken from online forum and two players, whose behaviour was taken from the existing AI bots found on the internet for Ticket to Ride. From here on, the forum player is referenced as card hoarding player and the two players from the internet are called as the rule-based player and the evaluation player.

5.1 MCTS Player

The Monte-Carlo Tree Search player was already implemented in the TAG framework. Multiple heuristic functions were created which enabled the MCTS player to analyze the game states and find the preferred actions. Before MCTS player was compared against other players the algorithm parameters and heuristic functions had to be tested to find the combination that gave the best results.

From the 4 players, MCTS is the only player whose decision making is directly affected by other players as when applying the algorithm the player has to simulate the taken actions of other players. Since Ticket to Ride is imperfect information game the actual actions that other players can take should not be known to the MCTS player. However, in the used MCTS player a simplification was made that the player knows exactly what actions others can take. The hidden train card and destination ticket decks were copied and shuffled before each simulation phase so the player would not know exactly what cards are given by choosing drawing hidden train card or destination tickets actions.

5.1.1 Parameters

The TAG framework enables to change many parameters for the MCTS algorithm. The main parameters used to tune the player were rollout depth, K, player budget, maximum tree depth and opponent tree policy.

The rollout depth parameter is used in the rollout phase of the algorithm. It determines how many actions at maximum are performed from the leaf node until the reached state is evaluated. The moves of the opponents also count against the rollout length. Since most of the player turns involve a player choosing an action two times then with a rollout length 20 in 4-player game on average only 3 three player's turns would be simulated as 6 actions would be taken by the player and 14 actions would be used to simulate the opponents turns.

After the rollout is finished the reached state is evaluated using the heuristic function and the change in the evaluation of reached state compared to the root state is backed up the tree. The framework supports two types of state evaluations, which is controlled by the opponent tree policy parameter. The first is called *SelfOnly* where when evaluating the state the heuristic function is only used to assess the state from the player perspective. The other option is called *Paranoid* where the state is also evaluated from the perspective of the opponents and the changes in opponents' scores are deducted from the player state value. The *SelfOnly* value was used in the experiments as in the actual game the player should not have access to the other players' information that is used in the heuristic functions to evaluate the state.

The K parameter is used when selecting new child to extend from the existing tree. It balances the exploitation and exploration of the tree. Maximum tree depth limits the tree, so that new child nodes are not created if this depth is reached. However, during testing

it came out that since the action space is relatively large the maximum reached depth in tree is very small and most of the time the tree does not reach more than depth of 10 and for this reason tuning this parameter was not considered further. For the experiments the max tree depth was increased to 100 to avoid the situation where at the end of the game due to lack of options the tree depth limit is reached.

The player budget limits the number of times the MCTS algorithm is applied. In TAG framework there are multiple budget types:

- Forward model calls - The number of times the forward model is used to advance the game state to new state using an action.
- Copy calls - The number of times that the game state is copied.
- Time - The player has to make a decision in the allocated time frame, if this is exceeded the player might have to skip his turn.

For the MCTS player used in the experiments the number of times the forward model is used to advance the game state was selected as the budget. The advantage of this over the time based budget was that in case the rollout length is smaller the state has to be copied more times which is a slow process and this reduces the number of times the created tree is visited compared to the MCTS player where rollout length is longer and the state is copied less times. Forward model based budget is more fair towards the players as they can advance the state similar number of times regardless of the time it takes. The moves of the opponents also count against the budget of the player. This means that after the player has taken an action it might take up to 6 forward model calls to reach the next player turn in case of a 4-player game.

The TAG framework also supports some of the enhancements to the MCTS algorithm like progressive widening and move-average sampling technique (MAST). Progressive unpruning, which is similar to progressive widening, proved to be successful when playing Ticket to Ride in the work done by Huchler [10]. However, during short experimentation with these modifications to the MCTS the results of the MCTS players did not improve and both progressive widening and MAST were not used in the final version of the MCTS player.

5.1.2 Heuristic function

Besides the parameters, the heuristic function is the second contributor to deciding which actions are taken. In total, 4 different heuristic functions were created. All of them return a value between -1 and 1, where -1 indicates that the player lost and 1 indicates that the player won. In case the game has not finished the 4 heuristics use different equations to assess the state.

The first heuristic (OnlyScore) was the most simple and only included the current score of the player when evaluating the state. It is presented in Equation 2. $playerCurrentScore$

is the current score of the player and it does not contain points from destination tickets or longest route as they are added at the end of the game.

$$playerCurrentScore/295.0 \tag{2}$$

The 295 is the maximum score a player can achieve in the base Ticket to Ride 2-player game and therefore it is guaranteed that the result of the equation is less than 1 [27]. To achieve a score of 295 the player must draw and finish almost all the destination tickets except 5 of them, which makes achieving 295 impossible in case of 4 players because it is guaranteed that other players have at least 6 tickets. The issue with this heuristic function was that the player preferred short term success over long term goals and was unable to finish the destination tickets handed out at the start of the game. Rest of the heuristic functions were created to try to guide the player into focusing on the routes needed for destination tickets.

The second heuristic function (DestinationProgress) considers the progress that the player has made towards to finishing a destination ticket in addition to the score of the player. Equation 3 shows how the state value is calculated by the second heuristic.

$$(playerCurrentScore + pointsFromDestinationTickets)/295.0 \tag{3}$$

$pointsFromDestinationTickets$ is a sum of all destination tickets values which were calculated by checking the progress the player has made with a destination ticket. If a ticket was finished the player was awarded the points from completing the ticket. Otherwise, the formula in Equation 4 was used to award points for the progress made towards finishing a destination ticket.

$$ticketPoints * (-1 + shortestPathOwnedPercentage) \tag{4}$$

In the equation the points awarded by completing the ticket is represented by $ticketPoints$. $shortestPathOwnedPercentage$ is calculated by finding the shortest path between the cities on the destination ticket and dividing the total length of the owned routes in the shortest path with the total length of all the routes in the shortest path. The shortest path is found the same way as for the rule-based player described in Section 5.3.

The idea behind the third heuristic (IncludeCards) was to include the train cards in player's hand in the state evaluation as well. With the second heuristic it did not matter if player had 1 card or 10 cards in hand and if the cards are needed for a route or not and this caused a lot of actions to not have an effect on the game state value. The third heuristic still uses the Equation 3 but Equation 5 is used for calculating $pointsFromDestinationTickets$ instead of the Equation 4.

$$ticketPoints * (-1 + shortestPathOwnedPercentage * 2 + neededCardsPercentage) \tag{5}$$

The $neededCardsPercentage$ is calculated by dividing the number of cards in player's hand that can be used to claim a route from the shortest path with the number of total cards needed to finish the missing routes in the shortest path. However, the heuristic created an issue where in case some of the longer routes are obtained and there are useful cards in the player's hand the player would keep drawing new destination tickets as the points added by the new destination ticket to the value of the game state would be bigger than 0. To prevent the player from endlessly drawing the destination tickets at later stages of the game an additional condition was added that when there are more than 3 unfinished destination tickets then the minimum from the result of Equation 5 and the minus points for the unfinished destination tickets was used.

The last heuristic (CollectAtFirst) was affected by one of the strategies that involves collecting cards at the beginning of the game. For the first 11 turns the state evaluation is only based on the cards that the player has. The used formula for evaluating the state in the first 11 turns is shown in Equation 6.

$$SUM(neededCardsPercentage)/nrOfDestinationTickets \tag{6}$$

The $neededCardsPercentage$ was calculated for all player destination tickets and the results were summed together and divided by the number of destination tickets. After turn 11 the state was evaluated using the third heuristic that considered the cards in player's hand as well.

5.2 Card hoarding player

The idea for how the player should behave came from an online forum [6]. It was suggested that collecting cards for the first 20-21 turns and then only buying the longest routes to finish the game quickly is advantageous in the game as other players do not have enough turns to finish destination tickets and are given minus points.

The player starts by selecting the two lowest cost destination ticket cards to minimize the minus points received in the end for not completing them. For the next 9 rounds the player always draws two cards from the hidden train cards deck. Starting from round 10, the players starts drawing cards from the visible train cards deck. A card with a color is drawn from the visible deck if the player does not have at least 5 train cards of that color. If there is not such color in the visible deck then the player draws from the hidden train cards deck. At the end of round 21 the player would have 46 (4 + 20 * 2) cards. Starting from the 22 game round the player would use those cards to buy the longest available routes on the map until there are not any cards in his hand anymore and the end of the game is triggered. In the best case scenario, the end game condition would be triggered after the player has bought 8 routes and the game would have ended by round 30.

5.3 Rule-based player

This player was created in one of the GitHub repositories that had implemented Ticket to Ride with graphical user interface in Java [28]. The player had to be rewritten to add it to the framework but the general algorithm behind the player decisions remained the same with slight modifications.

The modifications were caused by the differences in the implementations of the game. The version in GitHub had made some simplifications during the implementation. Player was allowed to draw only one card in his turn instead of the two cards. This limitation increases the value of drawing visible wild cards, because an additional card is not lost compared to the by the rules implementation where player gets to draw two cards if the cards are not wild cards from the visible train cards deck. Another simplification was that the player was unable to select new destination tickets after drawing them, instead if the draw destination tickets action was chosen the player was given the first card in the destination tickets deck.

Algorithm 1 Rule-based player action taking

```
1: Find obtainable train routes (longest first)
2: Calculate cost of destination tickets
3: if CostOfDestinationTickets = 0 then
4:     if CurrentGameRound < 30 then
5:         return Draw new destination ticket cards action
6:     end if
7:     if CanBuyRoute then
8:         return Buy the route action
9:     end if
10:    return Draw cards action
11: end if
12: for uncompletedDestinationTicket do
13:    if canBuyRouteFromTicketPath then
14:        return Buy route from path action
15:    end if
16: end for
17: if wildNotInVisibleDeck then
18:    Find next route to buy
19:    if neededColorInVisibleDeck then
20:        return Draw card with needed color action
21:    end if
22: end if
23: return Draw cards action
```

In the first turn the player selects two destination tickets randomly from the three given destination tickets. Pseudo-code of the algorithm for taking actions after turn 1 is provided

in Algorithm 1. The player keeps track of completed destination tickets and tickets that can not be completed anymore because an opponent has blocked the route. First the player finds the routes that it can buy and sorts them so that the longest routes are first as they give more points (Line 1). Then the list of completed and the unfinishable tickets are updated and the cost of the tickets that can be finished is calculated (Line 2). The cost of the tickets that can be finished is equal to the number of train carts needed to connect the cities on the tickets. The amount of needed train carts is equal to the length of routes in the shortest path between the two cities that are not owned by the player. To find a shortest path between the cities on the destination ticket the player uses breadth-first search[6].

In case all the player destination tickets are completed or can not be completed the player draws new destination tickets if the current game round is less than 30 (Line 4-6). Round 30 was selected as the limit after which to not draw new destinations as otherwise the player might not have enough turns to finish the new tickets. If the game round is bigger than 30 the player buys the longest available route or draws a card if no routes are available (Line 7-10).

In case the player has uncompleted destination tickets then for every destination ticket the player finds the shortest possible path for connecting the cities on the destination ticket and if the player can claim a route from that path it decides to do so (Line 12-16). If no routes are available to be bought from any of the paths that would connect the cities on the destination tickets the player checks if wild card is in the visible deck (Line 17). If there is not any wild train cards in the visible deck it checks if there is any card in the visible deck that is needed for any of the missing routes from the paths that would connect the destination tickets' cities (Line 18-19). If there was a needed card in the visible deck it takes it as the first card drawn in the turn and draws the second card from the hidden train deck (Line 20). Finally, if there was a wild card in the visible deck or none of the train cards is needed from the visible deck then the player decides to draw train cards (Line 23).

When the player decides to draw train cards (Line 10 and 23) then drawing a wild card from the visible train deck is preferred. Otherwise, the player draws from the hidden train cards deck. When the player is choosing which cards to use for claiming a route without a specific color then the color is used which has most cards in player's hand. When drawing new destination tickets the destination tickets that require less than 5 train carts to complete are kept or if there is no such ticket then the ticket which needs the least amount of train carts to finish is selected.

5.4 Evaluation Player

The algorithm for this player again came from one of the Ticket to Ride implementations from GitHub, which was originally written in Python [29]. Their implementation of the

[6] `https://en.wikipedia.org/wiki/Breadth-first_search`

game followed the rules described in the rule book of the original game. However, their player logic had an issue where the found path to connect all destination tickets was incorrect. The issue was fixed in the TAG framework implementation of the player.

Compared to the rule-based player it has some differences. Rule-based player used breadth-first search to find the shortest path between two cities, but this player tries to find all possible paths to the destination city that can be finished with 45 train carts. The paths are searched by starting from one city and then using all the routes from that city to see where they lead and so on with each new reached city. However, the path finding is done until 100 paths are found or the cities have been visited 10 000 times in total. If the 10 000 limit is reached it is also possible that no route has been found. In case it is the first turn of the game the path search between the cities is carried out until at least one path is found regardless of the number of iterations it takes. This is based on the assumption that in the first turn it is always possible to find a path that connects any of the two cities. The paths that connect the cities are sorted so that the shortest routes are first. Other big difference with the rule-based player is that when choosing the next action this player evaluates every possible action and chooses the action that has the highest score.

At the start of the game the player selects the kept destination tickets by finding the destination tickets combination that gives the most points and where the needed routes require less than 46 train carts. 46 train carts limit is used because player only has 45 train carts available. The needed routes for the combination of destination tickets are found by finding the possible paths between cities on every destination ticket and then finding the union of the combination of the destination tickets possible paths. The unions of the paths that require more than 45 train carts are discarded and the remaining are sorted so that the shortest in length are first. The combined paths are evaluated and the one with the smallest cost is selected for the destination tickets combination. The cost is determined by finding the missing train cards from the path where for routes without certain color the cost is equal to the length of such routes and for other routes the number of cards needed for every color is found and for every color the cost is equal to the number of cards needed squared. The path with smallest cost is compared to the least expensive paths of the other combination of destination tickets and the tickets to keep are chosen using the Equation 7. The combination of destination tickets that has the smallest value is kept and player remembers the paths to finish the selected destination tickets.

$$combinedPathCost - pointsFromCombinedPath * 0.5 \tag{7}$$

At every player turn the player checks if the existing best path to complete the destination tickets is still available. If it is available, all paths that can complete the tickets are re-evaluated based on the cards in player's hand. The same evaluation is used when evaluating the paths when selecting starting destination tickets where the number of cards needed of a color is squared and for routes without certain color the length of the route is the cost. If the best path is no longer available a new best path and other possible

paths are found similarly when finding paths for the destination tickets at the beginning of the game.

After updating the best path the player finds the cards and the routes that are needed to finish the path. If the player has no missing routes it deducts that the destination tickets are completed. To select an action the player evaluates every possible action and selects the one with the best score. The score for every action is given in Table 2. Besides the -1000 used in *KeepDestinationTickets* evaluation the constant values used in the formulas to score actions were defined in the original implementation.

Action	Action scoring formula
SelectRouteToBuild (part of path)	Equation 8
SelectRouteToBuild (not part of path)	-1
SelectRouteToBuild (tickets completed)	$pointsFromBuildingRoute$
DrawHiddenTrainCard	1
DrawVisibleTrainCard	$nrOfCardsNeededOfThatColor$
DrawWildTrainCard	2
DrawDestinationTickets (tickets incomplete)	-1
DrawDestinationTickets (tickets incomplete)	$-15 + trainCartsLeft$
BuildRoute (using wild cards)	$-9 * wildCardsUsed$
BuildRoute (wild routes)	$-cardsFromColorNeededForPathRoutes$
BuildRoute	0
KeepDestinationTickets (can not connect the cities)	-1000
KeepDestinationTickets	$-1 * pathCost$
DrawSecondHiddenTrainCard	1
DrawSecondVisibleTrainCard	$nrOfCardsNeededOfThatColor$
EmptyAction	0

Table 2: Rules for evaluating actions

Drawing new destination tickets is only considered if the player has already finished the existing destination tickets. The value of drawing new destination tickets action is related to the number of train carts the player has left. Therefore, to draw new tickets the player must have finished the initial tickets and have at least 16 train carts left because then the value of the action is equal to drawing a card from a hidden deck. The purpose of using train carts to evaluate the action is that it ensures that the player has enough train carts to potentially finish new tickets. Additionally, it ensures that the game should not be ending in the next couple of turns as usually the difference between the number of used train carts between the players is smaller than 15. After the player has decided to draw new tickets the *KeepDestinationTickets* actions are evaluated in the same way as at the start of the game but this time it is possible to keep only one ticket. For the actions that contain tickets that can not be completed -1000 is used to mark that it should not be taken. For other actions the cost is calculated the same way as at the start of the game but the result is multiplied by -1 to get the action with the cheapest path.

Drawing a card is similar to the rule-based player where drawing wild train card is preferred

over drawing from the hidden deck. Also, similarly to the other player if there is a needed card in visible deck it is preferred over drawing from the hidden deck. In case more than 2 cards from a color are needed and the color is available in visible deck it is also preferred over drawing a wild card.

Original implementation had claiming a route and choosing which cards to use as one action. In the player added to the framework it was split into two: evaluating the obtainable route and deciding which cards to use for it. Splitting into two different actions was necessary as they were separate in the implemented game. The evaluation of the obtaining routes actions depends on if there are any uncompleted destination tickets. In case there are not any destination tickets that have to be completed the value of the route is equal to the points it awards to the player using Table 1. Otherwise, if destination tickets are not finished and if the route is not part of the destination ticket path it gets a score of -1, because the player should not waste resources on routes that do not help to finish tickets. If the route is part of the destinations path then the Equation 8 is used. In the equation Table 1 is used for the $routePoints$ and $totalPathPoints$ is the points awarded by finishing all the routes in the path and $unbuiltRoutesPoints$ is the number of points awarded by the routes that are not yet built from the path.

$$2 + routePoints + totalPathPoints - unbuiltRoutesPoints \tag{8}$$

The decision on which cards to use to claim a route depends on two factors, if any wild cards are used and if the claimed route does not have a certain color. Every wild card that is used by the action is -9 points. For the routes without a specific color it is checked whether the used card is needed for any of the routes on the destination tickets path and the action is given minus points equal to the number of times it is used by the routes in the path. If no minus points are given from the used cards the *BuildRoute* action has a value of 0.

Splitting obtaining a route into two separate actions affected the behaviour of the player as when they are both in one action the minus points from using wild and needed cards would reduce the score of an action and allow the player to retain the needed cards. However, when the claiming of a route and using the cards are one action it can also lead to the player overvaluing the cards and allowing other players to buy the needed route instead.

The *KeepDestinationTickets* and *BuildRoute* actions both have negative score or at best are 0. This is possible because they are compared only to the same type of actions and the purpose is to minimize the loss to the player.

6 Experiments

The carried out experiments were divided into two categories: optimizing the MCTS player and comparing all players on various Ticket to Ride variants. All experiments were

carried out on the TAG framework and all games were played using 4 players.

6.1 MCTS player optimization

The MCTS player optimization consisted of optimizing the parameters of the algorithm and finding a heuristic that gave the best results. Unless stated otherwise all games were played with 3 MCTS players that used the default parameters from the framework and one player under test whose statistics were collected. The used default parameters are given in Table 3. All players were given equal budget, which was 8000 at first but was later changed to 20 000 forward model calls. All MCTS player optimizations were done on the Ticket to Ride USA map using the base game destination tickets.

Parameter	Value
Rollout length	10
K	sqrt(2)
Max tree depth	100
Opponent tree policy	SelfOnly

Table 3: MCTS default values in the optimization experiments.

6.1.1 Parameter optimization

The optimization of the parameters was done first using random grid search and the heuristic which did not include cards in player's hand but considered the progress towards destination tickets. The optimization focused on the rollout length and K as it was believed that these parameters have the most effect. The maximum and minimum values of the parameters for grid search are given in Table 4. At first the player budget was 8000 forward model calls and 100 games were played where the player under test played 25 games at each starting position.

Parameter	Minimum	Maximum
Rollout length	5	50
K	0	10

Table 4: Parameter search minimum and maximum values.

After the first test runs the parameters upper and lower bound were updated based on the achieved win rate of the player and the parameter range was reduced. The rollout length minimum value was changed to 20 and the K maximum value was set to 5 as first results showed that the performance suffered if the length was smaller or K was bigger. The player budget was also increased to 20 000 as the players struggled to finish the destination tickets. Due to the increased budget the number of played games was reduced to 20 because otherwise it would take several days to test one parameter combination. Therefore, the player under test would play 5 games at each starting location. The win rates of the player using the parameter combinations are displayed on Figure 10.

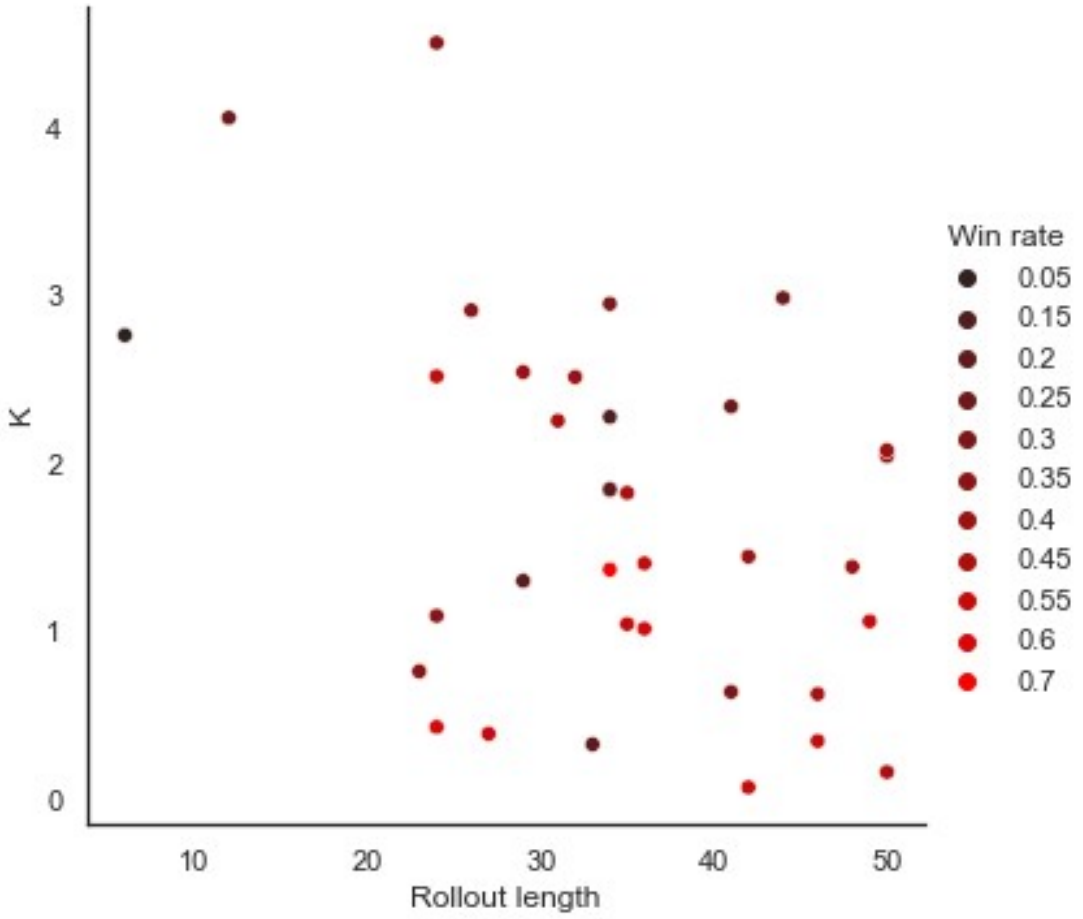

Figure 10: Results of the parameters grid search. Higher K values lead to worse win rate and higher rollout length parameter does not seem to have had an effect on the win rate.

This shows that there is a slight tendency that when K was around 1 the player achieved higher win rate. The figure also shows one experiment where the rollout length was 33 and K was 0.32 and the player managed to only win 20% of the games, which contradicts the general tendency that lower K values lead to higher win rate. The potential cause for the parameters to perform worse than other similar combinations is that the player with these values decided to keep or was given a little bit longer destination tickets that increased the minus points the player got at the end of the game as the player was unable to finish them. The rollout length parameter does not seem to have an effect on the player win rate. This suggests that the player does not learn additional information by simulating actions from the tree node. Also, longer rollout length results in using more forward model calls in one search iteration and this leads to fewer search iterations compared to the shorter rollout length. The results showed that rollout length 34 and K 1.361 had the highest win rate.

6.1.2 MCTS heuristic

The four heuristic functions described in the Section 5.1.2 played against each other in 100 games. Every player started 25 times at each starting position. They all used the best parameters found in the Section 6.1.1, rollout length 34 and K was 1.361. The player budget was limited to 10 000 to reduce the time it took to run the test. Drawing new destination ticket cards during the game was disabled to allow players to spend more budget on selecting between the other actions. The results are shown in Table 5.

Heuristic	Win rate	Average Points	Median Points	Destination tickets completed	Percentage of gained points
IncludeCards	0.34	35.78	35	0.24	25.62%
CollectAtFirst	0.29	39.97	36.5	0.6	24.97%
DestinationProgress	0.24	37.79	38.5	0.48	24.09%
OnlyScore	0.22	34.46	31	0.24	25.32%

Table 5: Heuristic function results. The completed destination tickets and percentage of gained points are averages over 100 runs. The heuristic that includes the value of cards in hand achieved the highest win rate but the heuristic which encouraged collecting cards at the start of the game managed to achieve higher average score.

Note that the win rate of the players do not add up to 1 because it is possible that multiple players have the same highest score and then all of them are awarded the win. The tie-breaker rules were not used in the current experiment. The gained points is the amount of points that the player earned during the game. The difference with the end score is that in the end score in case the player did not finish the destination ticket the points from the ticket would be deducted from the player score.

The heuristic that included the cards in player hand when evaluating the state had the highest win rate. However, when looking at the average points and destination tickets completed it does worse than the heuristic that did not include cards in evaluation and the heuristic that encourages the player to collect cards at the start of the game. The heuristic that includes cards seems to buy more routes on the map by the cost of not finishing destination tickets, this claim is backed by the fact that the percentage of gained points is highest among the heuristics but the average end score is smaller than for others. Although, the heuristic with only the progress towards destination tickets achieved the highest median points it was third in the win rate. This suggests that this heuristic was more stable than the other two heuristics that had higher win rate. Figure 11 shows the distribution of points scored by the different heuristic function players.

The scored points distribution is quite similar for all the heuristics. However, the minimum score by the heuristic that encourages collecting cards at the beginning is considerably higher than for the others. In addition, the maximum score is also higher than the maximum score achieved by the other heuristics. The figure also confirms that the heuristic which only awards the progress towards destination tickets was the most stable.

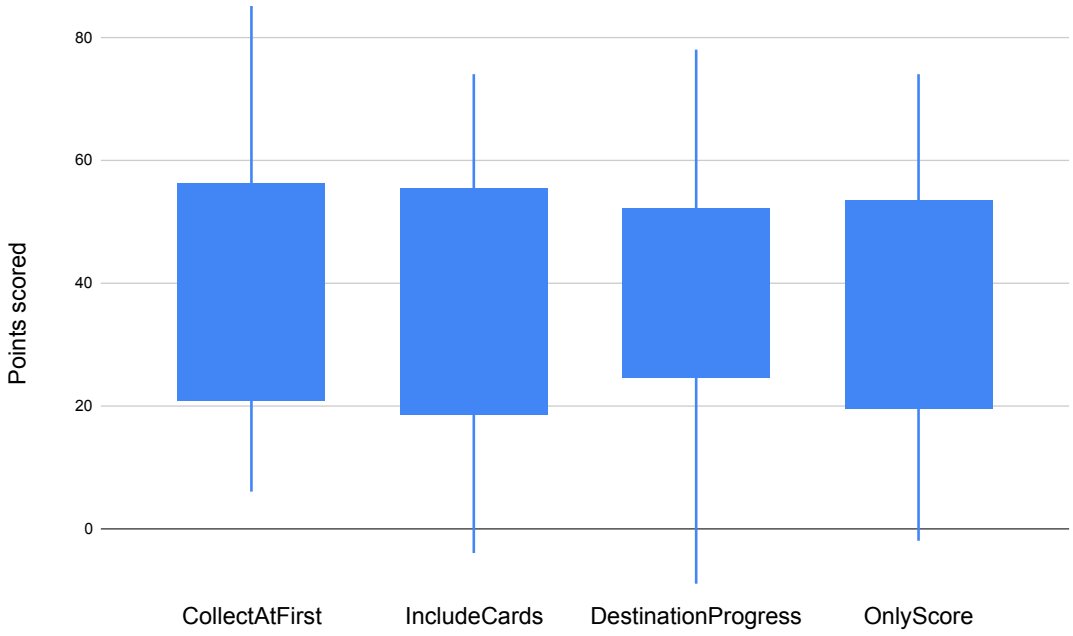

Figure 11: Scored points distribution by the 4 heuristic players. The distribution of points is quite similar for all heuristics, however the player who collected cards at the start managed to achieve higher maximum and higher minimum points than others.

Since the points distribution showed that all 4 heuristics achieved similar scores an additional experiment was made were the $CollectAtFirst$ and the $IncludeCards$ heuristics were put against each other because they had the highest win rates. Two players used one and two players used the other heuristic. One of the players used rollout length 34 and the other player using the same heuristic had rollout length as 0, which means that the player had no rollout step. In previous works skipping the rollout step has improved the performance of the MCTS in large actions spaces [30]. The players were given a budget of 20 000 forward model calls. The results are shown in Table 6. The used value for the rollout length is provided after the name of the heuristic. Note that the win rate does not add up to 1 due to the tie-breaker rules not being used.

The players who were encouraged to collect cards for the first 11 turns won less than the players who started claiming routes from the first turn. In addition, collecting cards for the first turns seems to have a negative effect on finishing destination tickets as the players who collected cards managed to finish less tickets on average. Interestingly, the increased player budget did not lead to more destination tickets being completed but the opposite. Based on the results the heuristic that awards claiming routes from the beginning of the game and awards player for drawing needed cards was selected as the best heuristic function. Before comparing MCTS against other players few tests runs with

Player	Win rate	Points	Destination tickets completed	Percentage of gained points
IncludeCards_34	0.35	46.68	0.28	26.27%
IncludeCards_0	0.29	42.41	0.24	24.89%
CollectAtFirst_0	0.24	35.81	0.18	24.35%
CollectAtFirst_34	0.22	35.94	0.24	24.51%

Table 6: Comparison of the two heuristic functions. Points, completed destination tickets and percentage of gained points are averages of 100 runs. The players with the heuristic that included cards managed to win more games than the collecting heuristic regardless of the used rollout length.

random parameters were done to decide the best rollout length and K combination.

Based on the results of the experiments the final MCTS player configuration was created, which was put up against other players. The budget was increased to 40 000 as MCTS is the only time-consuming player. The rollout length was set to 31 and the K was set to 0.229. In the rollout the actions to simulate were chosen randomly and the reached state was evaluated only from the player perspective using the heuristic that evaluated the player cards and awarded claiming routes from the beginning.

6.2 Comparing players

The 4 players described in Section 5 were put up against each other in the different variations of the Ticket to Ride game. The Europe, Germany and the USA map of the Ticket to Ride board games were used. In addition, the proposed destination tickets combinations in the Ticket to Ride: 1910 expansion for the USA map were also used in the experiments. The end of the game bonus points for the longest route or most destination tickets finished or both were used according to the written rules of the relevant map or destination tickets combination. All maps and destination tickets variation tests consisted of having the players play 100 games where each of them starts 25 times at one starting position. In case of a tie in scores, the winner was determined using the base game tie-breaker rules for all maps.

The first experiment had the 4 players playing the base game of the Ticket to Ride. At the end of the game, the player with the longest continuous route was given 10 extra points. The results are shown in Table 7. The player who collects cards for the first 20 turns achieved the highest win rate. Although the rule-based player managed to complete at least 1 destination ticket on average, it still did not manage to earn bonus points from the tickets. This could be caused by the game finishing relatively quickly as all the games were over by round 29 or 30 and the player did not have enough turns to finish the longer tickets, which give more points. Even tough the rule-based player got the most points for finishing tickets it had the lowest points average, which indicates that using the shortest paths to connect cities might not be the best approach on the USA map.

Player	Win rate	Points	Tickets completed	Total points from tickets
HoardCards	0.85	88.62	0	-18.96
Evaluation	0.14	44.31	0.8	-8.5
MCTS	0.01	33.49	0.35	-17.11
RuleBased	0	32.03	1.51	-2.17

Table 7: Comparison of the players in the base game. Points, completed tickets and total points from destination tickets are averages over 100 runs. HoardCards player won by far the most games. The rule-based player managed to completed the most tickets but had the smallest average score.

The next experiment was to assess the players on the other two maps. In case of Europe the player with the longest route got extra 10 points at the end of the game and in case of Germany the player who finished the most destination tickets was given extra 15 points. Table 8 shows the results of the players on the Europe and Germany maps. The card hoarding player still won most of the games on both maps but the win rate was smaller and in Europe the number of points was on average almost 10 points smaller than on the USA. The potential cause for getting less points on Europe is that that map only has 3 routes which are with length 6 or longer and other routes are with length 4 or less. This reduces the number of points awarded by claiming the routes compared to the USA and also increases the number of turns it takes to spend all train carts, which gives more time to other players to finish their tickets. The evaluation and rule-based players managed to win about equal number of games on the both maps, although the rule-based player averaged more points, this suggests that the evaluation player is more ambitious and takes bigger risks with the path it finds and this leads to either a win or a bad score.

Map	Europe				Germany			
	Win rate	Points	Tickets completed	Total from tickets	Win rate	Points	Tickets completed	Total from tickets
HoardCards	0.79	74.67	0.02	-15.36	0.8	82.19	0.1	-14.64
RuleBased	0.1	47.59	2.49	9.55	0.09	49.51	2.11	8.61
Evaluation	0.08	37.87	1	-6.43	0.1	35.53	0.96	-8.23
MCTS	0.03	39.37	0.3	-16.48	0.01	34.51	0.47	-14.77

Table 8: Results of the players on the Europe and Germany maps. HoardCards was still the most successful player but the win rate was smaller than before. Rule-based and evaluation player managed to win about equal amount of games on both maps.

Thirdly, the players were tested on the map of the USA, but the destination tickets were changed according to the three variants described in the rules of the Ticket to Ride: 1910 expansion to the base game. The results are shown on Table 9. The card hoarding player still managed to win most of the games, but on the variant were destination tickets with Big Cities were used it achieved its lowest win rate. Compared to the original game the different variants did not affect the points scored by the card hoarding player. However,

the points scored by the other players increased in two of the variants and the minus points from the destination tickets was reduced which suggests that the rules of the variants help other strategies against the card hoarding but it is still not enough to beat it on regular basis. The main area of improvement for the other strategies was that they were able to complete more destination tickets on average and get more points from them. For the rule-based player the improvement in points is mostly because the player with most finished destination tickets was given 15 extra points at the end of the game in case of Big Cities and 1910 destination tickets.

Variant	1910			Mega			Big cities		
	Win rate	Points	Tickets completed	Win rate	Points	Tickets completed	Win rate	Points	Tickets completed
HoardCards	0.86	90.42	0.01	0.82	84.97	0.01	0.75	87.95	0
Evaluation	0.13	48.27	1.04	0.08	28.97	0.58	0.19	50.52	0.97
RuleBased	0.01	49.3	2.01	0.04	27.91	1.31	0.06	49.13	1.97
MCTS	0	36.12	0.51	0.06	42.22	0.6	0	37.7	0.42

Table 9: Results of the players on the expansion destination tickets variants. Regardless of the destination tickets, the HoardCards player won most of the games.

As shown by the experiments the card hoarding player is most successful regardless of the used map and the destination tickets combinations. One of the causes is that the longest routes are overvalued as shown by Witter and Lyford [5]. They proposed a changed scoring system where the points awarded for a route with length k is equal to $a * k$, where a is a value between 3.5 and 5.5. The proposed scoring system was tested with the players on the map of the USA using the base game destination tickets and 4.5 was used as the a constant for calculating points from a route. The results are shown on Table 10. The change in scoring did not balance the win rates of the players but instead it had a reverse effect where the card hoarding player got even more dominant. The average points scored by the other players were quite similar, which suggests that the change might balance the other approaches. The downside of the changed scoring is that the points from destination tickets was not changed and the value of completing a ticket is much smaller.

Player	Win rate	Points
HoardCards	0.98	185.24
Evaluation	0.02	111.79
MCTS	0	111.89
RuleBased	0	113.06

Table 10: Results of the players with the changed scoring system. The change in scoring instead of making the game more equal actually increased the advantage of the HoardCards player strategy.

Additional games were played without the card hoarding player due to player being so dominant and the games being finished in less turns than usual, which does not give info

about the success of other strategies. The other 3 players played 100 games on the USA, Europe and Germany map. A filler player was used as the fourth player in those games. The filler was either rule-based, evaluation, MCTS or a player who took random actions and each type was used 25 times. The results for the 3 players are shown on Table 11. The win rates of the players do not add up to 1 as the fourth player which was used as the filler also managed to win some games. MCTS player struggled to complete destination tickets on all three maps, interestingly it managed to win more games on Europe than the rule-based player. The map of Germany suited the rule-based player the best as it almost won half of the games. The reasons why the Germany fitted the rule-based player the best are that extra points were given to the player who finished most tickets and that there are lot of short routes in Germany which reduces the effect of not claiming long routes. Although, rule-based player completed on average the most tickets on every map it did not lead to winning more on the map of USA and Europe, which suggests that on those maps it is not advantageous to complete destination tickets by claiming routes from the shortest path between the cities and other approaches should be preferred. Overall, when combining the results of the three maps the evaluation player was the most successful.

Map	USA			Europe			Germany		
	Win rate	Points	Tickets completed	Win rate	Points	Tickets completed	Win rate	Points	Tickets completed
Evaluation	0.49	84.72	1.68	0.35	66.86	1.71	0.32	78.4	1.9
RuleBased	0.25	72.48	2.23	0.2	65.9	2.76	0.45	86.62	2.96
MCTS	0.09	16.19	0.43	0.29	21.55	0.48	0.07	10.9	0.9

Table 11: Results of the players without the card hoarding player. The Germany map fitted the rule-based player the best as it managed to win almost half the games but the evaluation player was most successful on the other two maps.

6.2.1 AI players vs human

The created players were also compared against human players. The MCTS player was excluded from the games with a human player cause it takes about 10 seconds to finish its turn and we did not want to have the participants to wait for their turn while playing. In total there were 8 participants who all played one game of Ticket to Ride with 4 players where the other three players were card hoarding, evaluation and rule-based player. All games were played on the USA map with the base game destination tickets. No participant had played Ticket to Ride with the USA map before and for half of the participants it was the first time playing the board game. The results of the human and AI players are shown in Table 12. Card hoarding player won 6 games out of 8 and had the highest average score, which was expected as any of the human players were aware of the card collecting strategy and did not know to counter it. Human players were able to win 2 games and in both cases they were by one point and the paths of the human player's destination tickets overlapped in huge part, which suggests that to beat the card hoarding player the player

must get lucky with the destination tickets. The evaluation player averaged some points fewer than the human player and in most games it finished in third a couple of points behind the human player. All participants decided on which routes to claim based on their destination tickets and did not consider buying routes which did not help to complete a ticket unless it was in the later stages of the game when it was clear that the existing tickets can not be completed. This was also the reason why the player controlled by the participants did not manage to win more games as the claimed routes were often short and they were unable to complete the longer tickets as card hoarding player triggered the end of the game too fast.

Player	Win rate	Points	Tickets completed	Total points from tickets
HoardCards	0.75	92.13	0	-16.5
Human	0.25	62.38	2	7.25
Evaluation	0	55.25	1.5	6.38
RuleBased	0	31.13	1.38	-3.13

Table 12: Comparison of the AI players and a human player in the base game. Points, completed tickets and total points from destination tickets are averages over 100 runs. Card hoarding player won most games. Human player was able to win couple of games but on average the performance was similar to the evaluation player.

To have an idea on how MCTS would perform against human players we played 5 games ourselves against MCTS, rule-based and evaluation players. Card hoarding player was excluded to have 4 players in the game and as shown previously by the participants, it was able to beat the human player quite regularly. In those games we tried to play similarly like the participants did by focusing on completing the destination tickets. 3 out of those games were won by evaluation player and 2 were won by us. In those games MCTS achieved the 3rd place 4 times and was fourth in one game. In 4 games MCTS player managed to complete at least one destination ticket and in all the games it had used almost all of the 45 train carts. This suggests that the player does not focus on obtaining a certain route or path like the rule-based and evaluation approach but decides based on the available resources how to use them and maximise the received value.

7 Discussion

The conducted experiments showed that collecting cards at the start and then using them to obtain the longest routes is the best strategy regardless of the used map and the destination tickets. The other players had much less points on average, but were able to win some games when they managed to complete the given destination tickets before the game ended. The modifications introduced by the Ticket to Ride: 1910 expansion reduced the gap between the best player and others but it was not enough to balance the used strategies of the players. Although, the results suggest that card hoarding was dominant in the experiments the effectiveness of the strategy suffers if there is more

than one player doing it as then there is bigger competition for the routes that give more points and this leads to obtaining more shorter routes and gives more time to other players to finish their tickets. The modified scoring system proposed by Witter and Lyford [5], which should have made the game more equal for different strategies and reduce the value of the longer routes had an inverse effect as it improved the win rate of the best player in the conducted experiment and increased the points gap with the other players. One reason for the increased gap is that other strategies focus on completing destination tickets but with the modified scoring the actual value of completing a destination ticket is much smaller because on average the destination ticket gives 11 points, which with the modified scoring system is equivalent of claiming one route with length 3.

The experiment without the card hoarding player showed that the second best player was the player that combined the possible paths of the destination tickets and evaluated each action before choosing which to take. The player who claimed routes only from the shortest path of each destination ticket managed to complete the most tickets but since the routes used to complete them were short and gave few points it did not lead to a high score. MCTS player was the worst of all the players, although it managed to win more games than the rule-based player in some of the experiments but the average scored points was still lower than for the other players.

In the games with human player, the card hoarding player still managed to win the most which indicates that the strategy is viable in real-life games. In the games played without the card hoarding player the evaluation bot managed to beat the human a couple of times which suggests that the player might be challenging for the humans who play Ticket to Ride for fun. The rule-based and MCTS AI bots failed to consistently outperform the humans even tough half the games were played against players who played Ticket to Ride or Ticket to Ride with the USA map for the first time.

Although MCTS has achieved promising results in other implementations of board games [31, 32] it failed to achieve good scores while playing the implemented Ticket to Ride version. One of the reasons why the MCTS was not successful was that it was not able to finish destination tickets as it prioritised short term rewards over the long term goals. The performance of MCTS could possibly be improved by using different heuristic function. The used heuristic was based on finding the shortest path between each destination ticket which, as shown by the results of the rule-based player, is not a good approach to the game.

Since the logic behind three of the four players was specific to Ticket to Ride it is difficult to assess if the results of this thesis could be used for other games. Pan Am[7] and Thurn and Taxis [8] are an example of some of the games, which also include connecting points on the map and end game rewards and for such games the findings of this thesis could possibly be transferred.

[7] https://boardgamegeek.com/boardgame/303057/pan-am

[8] https://boardgamegeek.com/boardgame/21790/thurn-and-taxis

7.1 Future works

There are many possibilities for future works based on this thesis. Firstly, the players used in the experiments can be improved further as the current player types were based on the existing bots in other projects which had slightly different implementation of the game. For the rule-based player some additional optimizations are possible for deciding which cards to draw and when to draw new destination tickets. The evaluation player can also be improved by testing it with other values for certain actions and by improving the path finding of the two cities. Without including any board analysis and probability theory it should be possible to make the rule-based and evaluation player to be as good as an average Ticket to Ride player. With the addition of board and game state analysis the rule-based and evaluation players should be able to compete with the top-level human players.

As mentioned before MCTS has given good results on other board games and Huchler has showed that the MCTS player can handle the difficulty of Ticket to Ride in 2 player games [10]. However, in the conducted experiments with 4 players the used MCTS player struggled to compete with the other players. Potential improvements to help the MCTS algorithm to achieve better scores include improving the heuristic function and adding enhancements to the algorithm. Some possible improvements to the algorithm would be to add tree pruning as it allows to spend more resources on promising actions. Besides enhancing the algorithm the MCTS simulation step should be changed so that it does not use information that it does not have access to but instead deducts the possible game state based on the available information.

The map layouts besides the USA were used without the game and map specific rules. The Europe map has special routes which require certain number of wild cards or routes where three additional cards are drawn from the deck which determine if the cost of the route is increased or not. The Germany map has an additional way to get points by collecting passengers which are given on first come first served basis. In addition, the destination tickets were divided into two groups in the actual board game version of these maps but they were combined into one deck in the implementation. It would be interesting to see how the strategies are affected if the maps are used with their corresponding rules.

It would be interesting to see how agents that use machine learning would perform in Ticket to Ride. There are a couple of attempts that tried to investigate this area [19, 33] but neither of them managed to give an indication how successful machine learning would be. Strömberg and Lind used reinforcement learning agents on a simplified version of Ticket to Ride and the agents were able to mostly beat the player who took random actions in 2-player games but there is no information about how the reinforcement learning player would do against a human or rule-based player.

8 Conclusion

In this thesis, the digital version of the board game Ticket to Ride was created and analyzed. The Ticket to Ride implementation was added to an existing framework, TAG, which specialises in developing modern board games for AI research. The implemented version of the Ticket to Ride was used to assess 4 players on various editions of the game. The behaviour of the two players was taken from other existing Ticket to Ride projects, one player was based on the strategy suggested by the game community and the fourth player used MCTS to decide which actions to take. In total, 3 different maps and 3 destination ticket combinations proposed in the expansion to the game were used to evaluate how the performance of the players was affected.

The answer to the first research question, *What is the best performing Ticket to Ride AI player in 4 player games?*, based on the conducted experiments is that the player which was based on the strategy suggested by the community performed the best as it managed to win most of the games. The other players struggled to complete their strategic goals as the game finished too quickly. In the games where the community strategic player was excluded the player which evaluated every action before making a decision performed the best. Although previous works suggested that MCTS can achieve good results in case of 2 players then in most of the experiments conducted as part of this thesis the MCTS player struggled to compete with the other players. The potential causes for MCTS not performing as good as shown previously can be the difference in the used heuristic functions and that the algorithm used in this thesis did not use any tree pruning or other algorithm enhancements.

The answer to the second research question, *How the performance of the players is affected when using different Ticket to Ride variants?*, is that the community strategic player outplayed all other players regardless of the map and the used destination tickets. The player managed to win at least 75% of games on every tested configuration. Although the variations did not affect the winner most of the times, in some cases it still managed to balance the game more so that other players reduced the point gap to the winner and managed to win some games more. The Europe map was a good example as there are not that many long routes which reduced the amount of points the community strategy player got. Although, the game variations did not affect the best player it came out that for other players certain maps fit better for their strategies. An example of such variation is a map of Germany, where it was advantageous to finish destination tickets by obtaining the shortest path between the required cities but in case of other maps claiming the shortest path did not lead to a high score.

References

[1] G. Engelstein and I. Shalev, *Building Blocks of Tabletop Game Design: An Encyclopedia of Mechanisms*. CRC Press, 2019.

[2] Days of Wonder, "Ticket to Ride." https://www.daysofwonder.com/tickettoride/en/usa/, 2004. [Online; accessed 6-January-2022].

[3] Alexiane Achard, "Ticket to Ride World Championship 2014." https://www.daysofwonder.com/en/press-release/096/#newsstart, 2014. [Online; accessed 8-March-2022].

[4] Boardgaming World, "World Boardgaming Championships." http://www.boardgamers.org/eventhistory/ttr.html, 2020. [Online; accessed 8-March-2022].

[5] R. T. Witter and A. Lyford, "Applications of Graph Theory and Probability in the Board Game Ticket to Ride," in *International Conference on the Foundations of Digital Games*, FDG '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[6] Joe Jarvis, "What are the best strategies for Ticket to Ride?." https://www.quora.com/What-are-the-best-strategies-for-Ticket-to-Ride, 2018. [Online; accessed 12-March-2022].

[7] R. D. Gaina, M. Balla, A. Dockhorn, R. Montoliu, and D. Perez-Liebana, "TAG: A Tabletop Games Framework," in *Experimental AI in Games (EXAG), AIIDE 2020 Workshop*, 2020.

[8] D. Berwanger, L. Kaiser, and B. Puchala, "A Perfect-Information Construction for Coordination in Games," in *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)* (S. Chakraborty and A. Kumar, eds.), vol. 13 of *Leibniz International Proceedings in Informatics (LIPIcs)*, (Dagstuhl, Germany), pp. 387–398, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011.

[9] J. C. Harsanyi, *Games with Incomplete Information*, pp. 43–55. Dordrecht: Springer Netherlands, 2001.

[10] C. Huchler, "An MCTS agent for Ticket to Ride," Master's thesis, Maastricht University, 2015.

[11] R. Coulom, "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," vol. 4630, 05 2006.

[12] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo Tree Search: A New Framework for Game AI," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 4, pp. 216–217, Sep. 2021.

[13] James Goodman, "Monte Carlo Tree Search Parameters." https://github.com/GAIGResearch/TabletopGames/wiki/Monte-Carlo-Tree-Search-Parameters, 2021. [Online; accessed 10-June-2022].

[14] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[15] H. Finnsson and Y. Björnsson, "Simulation-Based Approach to General Game Playing," vol. 1, pp. 259–264, 01 2008.

[16] G. Chaslot, M. Winands, H. Herik, J. Uiterwijk, and B. Bouzy, "Progressive Strategies for Monte-Carlo Tree Search," *New Mathematics and Natural Computation*, vol. 04, pp. 343–357, 11 2008.

[17] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *Learning and Intelligent Optimization* (C. A. C. Coello, ed.), (Berlin, Heidelberg), pp. 433–445, Springer Berlin Heidelberg, 2011.

[18] F. de Mesentier Silva, S. Lee, J. Togelius, and A. Nealen, "AI-Based Playtesting of Contemporary Board Games," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, FDG '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[19] D. Dinjian and C. Nguyen, "The Difficulty of Learning Ticket to Ride."

[20] jkornel44, "Ticket-to-Ride-Europe." `https://github.com/jkornel44/Ticket-to-Ride-Europe`, 2021. [Online; accessed 10-May-2022].

[21] Days of Wonder, "Ticket to Ride: Europe Rules." `https://ncdn0.daysofwonder.com/tickettoride/de/img/te_rules_2015_en.pdf`, 2015. [Online; accessed 5-April-2022].

[22] Whizkid, "BoardGameGeek: Germany Tickets." `https://boardgamegeek.com/filepage/143266/germany-tickets`, 2017. [Online; accessed 5-April-2022].

[23] Amanda Zimmer, Bartłomiej Predki, "BoardGameGeek: Difference between TTR Germany and TTR Deutschland + 1902." `https://boardgamegeek.com/thread/1843369/difference-between-ttr-germany-and-ttr-deutschland`, 2017. [Online; accessed 5-April-2022].

[24] Days of Wonder, "Ticket to Ride: Germany Rules." `https://ncdn0.daysofwonder.com/tickettoride/de/img/tg_rules_2017_en.pdf`, 2017. [Online; accessed 5-April-2022].

[25] Paul Street, "Ticket to Ride USA 1910 Ticket Values and City Frequencies.doc." `https://boardgamegeek.com/filepage/23977/ticket-ride-usa-1910-ticket-values-and-city-freque`, 2007. [Online; accessed 5-April-2022].

[26] R. D. Gaina, M. Balla, A. Dockhorn, R. Montoliu, and D. Pérez-Liébana, "Design and Implementation of TAG: A Tabletop Games Framework," *ArXiv*, vol. abs/2009.12065, 2020.

[27] Adam Lauer, "Score Maximization Solution: 290." `https://boardgamegeek.com/filepage/40819/score-maximization-solution-290`, 2012. [Online; accessed 17-January-2022].

[28] Shao Sun, "AI-TicketToRide." `https://github.com/ss2cp/AI-TicketToRide`, 2019. [Online; accessed 15-March-2022].

[29] Steve Schwarcz, Jeffrey Twigg, Di Zeng, "Ticket To Ride Libary." `https://github.com/willdzeng/ticket_to_ride`, 2016. [Online; accessed 22-March-2022].

[30] D. Perez-Liebana, Y.-J. Hsu, S. Emmanouilidis, B. Khaleque, and R. Gaina, "Tribes: A New Turn-Based Strategy Game for AI Research," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, pp. 252–258, Oct. 2020.

[31] P. Ciancarini and G. P. Favini, "Monte carlo tree search in kriegspiel," *Artificial Intelligence*, vol. 174, no. 11, pp. 670–684, 2010.

[32] R. D. Gaina, J. Goodman, and D. Perez-Liebana, "TAG: Terraforming Mars," in *Proceedings of the 17th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2021.

[33] L. Strömberg and V. Lind, "Board Game AI Using Reinforcement Learning," Bachelor's Thesis, Örebro University, 2022.

A Ticket to Ride: Europe map

B Ticket to Ride: Germany map