Artificial Intelligence Approaches to Build Ticket to Ride Maps

Iain Smith, Calin Anton
MacEwan University, Edmonton, Alberta, Canada
smithi23@mymacewan.ca, antonc@macewan.ca

Abstract

Fun, as a game trait, is challenging to evaluate. Previous research explores game arc and game refinement to improve the quality of games. Fun, for some players, is having an even chance to win while executing their strategy. To explore this, we build boards for the game Ticket to Ride while optimizing for a given win rate between four AI agents. These agents execute popular strategies human players use: one-step thinking, long route exploitation, route focus, and destination hungry strategies. We create the underlying graph of a map by connecting several planar bipartite graphs. To build the map, we use a multiple phase design, with each phase implementing several simplified Monte Carlo Tree Search components. Within a phase, the components communicate with each other passively. The experiments show that the proposed approach results in improvements over randomly generated graphs and maps.

Introduction

Fun has no current evaluation, but ways of evaluating the quality of games have been developed and explored. Game refinement (Sutiono et al. 2014) and game arc (Silva et al. 2018) have been used to measure the quality of games based on the number of available options. These measures do not consider the quality of available moves; they strictly count the number of moves. Both these measures fail to consider that players may prefer different kinds of moves. They also do not consider if none of the remaining moves correspond to a player's strategies. To overcome these limitations and avoid evaluating the quality of every move, we consider a simplified measure of quality. Our strategy is to start with the desired win rate among several player agents then build games that produce a win ratio close to the desired one when simulated. To make games evenly fun, we try to balance the win ratio between different strategies of Ticket to Ride players. By balancing win rates, we design games that give players an even opportunity to execute their strategy and win. We assume that each player agent executes only its chosen strategy. Our newly designed game allows each player to perform several moves they deem quality moves and still win an even number of games. This evaluation would mean human players have the chance to follow their selected strategies and produce an engaging game in the process.

Ticket to Ride and Its Rules

Figure 1. Original USA Ticket to Ride board.

Ticket to Ride (TTR) is a board game for 2-5 players in which players build railway routes between cities and complete destinations for points. The game board comprises cities, the connections between them, a deck of colored train cards, and a destination card deck. Cities can be connected by either single or double routes. At the beginning of the game, players draw three destination cards and must keep at least two. Destinations depict two cities to connect and points gained from connecting those cities by building routes between them. If a player fails to connect the cities of a destination, that player loses these points. Players build routes between connected cities by spending train cards. The train cards are drawn from the top of the train deck or can be picked from five face-up cards. Players may pick two train cards or a locomotive that stands in for any color; however, they may only take that one card if they draw a face-up locomotive. Colored train cards are collected and spent along with a player's own trains to build routes of length one to six.

Players get points for each route, according to their length. Longer routes earn more points than shorter ones. Routes with any color other than grey must be built using trains cards that match the route color while grey routes may be built using any same-color train cards. At any time, instead of picking train cards, a player may choose to draw

three new destination cards, of which it has to keep at least one.

Players begin the game with 45 trains to use when building routes. The game ends one round after any player has two or fewer trains remaining. At the end of the game, players claim points from the destinations they completed, the routes they built and can claim extra points from having the longest path of trains. If the routes' lengths and colours are removed, the game board becomes a multi-graph with either single or double edges. If double edges are removed, the multi-graph becomes a simple graph which we will refer to as the underlying graph of a game board. The destinations, the double routes, the colours, and the lengths of the routes can thus be viewed as features added to the underlying graph.

Previous Work

Silva et al. explored a genetic algorithm to build maps for Ticket to Ride (Silva et al. 2018). They investigated underlying graphs which are valid under some restrictions for a TTR board. The fitness function evaluates the game arc, which quantifies the number of available moves throughout a game. The game arc they attempt to match implies a few decisions at the beginning. The number of decisions available into the mid-game gradually increases. Then it decreases to just a few decisions towards the end of the game.

Using the same engine and agents as Silva et al. Witter and Lyford (Witter and Lyford 2020) applied graph theory and probability tools to explore improvements to Ticket to Ride. They found that longer routes were disproportionately overvalued, which could be exploited by some players and proposed a new method for scoring to correct this issue. To improve the scoring mechanism, their method linearly searches modified route values to even the win rates between agents on the original USA map.

Player Strategies

Players may use any strategy they desire within the limits of Ticket to Ride's game rules. The information available to the players is limited because TTR is only partially observable: train cards and destination cards are hidden from other players. Without complete information, game agents that implement general multiplayer strategies like best-reply search, paranoid, and max^n (Schadd and Winands 2011) are at a disadvantage. For these reasons, we only use agents that implement the following popular strategies used by human players of TTR.

One Step Thinker Agent (OSTA)

The one step thinker agent simulates players that make decisions turn by turn. This agent selects new destinations only after completing all the current destinations it has. It executes a common strategy of some players by prioritizing expensive destinations and looking to complete the destinations as quickly as it can from its current position (Silva et al. 2017).

Path Agent (PA)

The path agent executes a similar strategy to the one step thinker agent but focuses on scoring on longer, more valuable routes. Often it works on routes that simultaneously complete multiple destinations. This agent never chooses to pick more destinations. After completing its original destinations, it focuses exclusively on the highest value routes (Silva et al. 2017).

Long Route Junkie Agent (LRJA)

The long route junkie agent implements a strategy that never claims low value short routes. It attempts to complete all the destinations it selected by completing only routes of length greater than three. By doing so, it typically loses points from destinations but makes up for them through the high value of the long routes it built (Silva et al. 2017).

Hungry Agent (HA)

The hungry agent uses a strategy focused on claiming as many points as possible from destinations by taking destinations until it has enough to spend all its trains on them. Then it formulates a plan to complete all these destinations while claiming the most important routes first (Silva et al. 2017).

Map Building Strategy

The overall approach we use is described by:

1) Start with the desired win distribution among players
2) Select or generate an underlying graph
3) Add double routes
4) Add destinations cards
5) Add lengths to routes
6) Add colors to routes

Steps 3, 4, 5, and 6 optimize the corresponding features towards the desired win distribution. The graph generating process in step 2 tries to optimize the graph selection with respect to the desired win distribution. Ideally, the optimization process will consider all extensions at once, but this would create a very large search space. By running the optimizations separately, we drastically reduce the search space. Steps 3 and 4; and 5 and 6, respectively, run separately but are synchronized by exchanging information after each choice. This further reduces the complexity of the optimization problem. We build maps using a multi-phase design inspired by the work of Abukhait et al. on Nine Men's Morris (Abukhait et al. 2019). The primary

difference is that we apply this design to a game **building** agent rather than a game **playing** agent.

The Distance Function

$$distance = \frac{\sum_{i=1}^{n}|d_i - c_i|}{2 - 2min_{i=1}^{n}(d_i)}$$

To measure the accuracy of the results, we introduce a distance between the desired win ratio and a win ratio calculated by simulating games on the tested board. The distance can be applied to any number of players (n). In our simulations, we fixed n to 4 as we experimented with four different strategies. The distance estimates the sum of the absolute error between the desired and the calculated win ratios. The sum is normalized by dividing it by the sum of the absolute error of the worst-case scenario, which happens when the agent with the smallest desired win ratio wins all the games. The value of the absolute error of the worst-case scenario is $2 - 2min_{i=1}^{n}(d_i)$, where d_i is the desired win rate for agent i, and c_i is the calculated win rate for agent i. After normalization, the distance will have values between 0 and 1, where 0 means a perfect match between the desired and calculated win ratio.

Consider, for example, a desired win ratio d of (0.35,0.4,0.25). If the calculated win rate c is (0.34,0.38,0.28), then the distance formula's numerator will be 0.01+0.02+0.03=0.06. For this d, the worst-case scenario is when c is (0,0,1). In this case, the denominator of the distance formula becomes 0.35+0.4+0.75, which is the same as 2-2*0.25. Thus, the distance is 0.06/1.5=0.04.

All optimizations try to minimize this distance.

We use several phases to build an optimized game board. (see Figure 2) Within a phase, optimizers can communicate with each other by sharing a common partial assignment. The purpose is to reduce the possibility of making changes that cause sudden shifts in the distance due to a sharp increase in the win rate of one of the agents.

We designed five optimizers; one used to generate an underlying graph – the **graph optimizer (GO)**, and four that optimize the features of the board: color – **the route color optimizer (CO)**, length-**the route length optimizer (LO)**, double routes – **the single-double route optimizer (SDO)**, and destination card value -**the destination value optimizer (DO)**. Each optimizer implements a simplified Monte Carlo Tree Search (Coulom 2007). The goal is to generate a full assignment of features that produces a game board on which players have a win ratio close to the desired one. We estimate the win ratio of a game board as the average of 20 simulated games between the player agents.

Graph Optimizer

The primary requirement of physical boards of TTR is that the underlying graph must be planar – it can be drawn without any intersections over edges.

Our graph optimizer starts by creating components of the graph – which we refer to as subgraphs, using plantri (Brinkmann and McKay 2007), a C program designed to produce unique planar graphs. It takes as parameters the order of the graph and, optionally, a range for the number of edges of the graph. Using plantri GO produces graphs with 12 vertices, 25 to 30 edges, and a minimum degree of 2. These numbers were chosen such that the final underlying graph has 36 cities, between 93 and 108 routes (some of which will become double routes in the next phases), and no end-of-the-line cities.

It starts by sampling from the first 10,000 graphs generated by plantri. It filters out graphs having a vertex of degree larger than 6 (this ensures that no city will have more than 7 routes in the final map, which is a feature of the original USA board). Thus, it reduces the number of candidates to only 86.

Using a procedure similar to the optimization presented in the next subsection, it finds three subgraphs that produce the closest win ratio to the desired one. It then connects them to create the final underlying graph. It adds 6 edges to each pair of subgraphs by repeating twice the following procedure:

1) Select from each graph the pair of vertices with the lowest degree

2) Add 4 edges by connecting each vertex from a pair to the vertices from the other pair

3) Remove the edge which connects the highest degree vertices from each pair.

The main benefit of this procedure is that it creates planar graphs and balances the degree sequence of the resulting graph.

Two-Phase Optimization Design

The optimization proceeds in two phases: one in which the destination card values and double routes are added; and one in which route colors and route lengths are added. The reason for this two-stage approach is that destinations cards and double routes complement each other, and so do route colors and lengths. For example, if there are fewer double routes, a city is more likely to be cut off. This will negatively influence the players holding destination cards that contain that city. The main reason for separating the phases is to reduce the search complexity. Another reason is that changes in features from one phase may result in sudden swings in features from the other phase. The two-phase approach avoids this by splitting the objective of the optimization between classes of features.

Each optimizer implements a simplified Monte Carlo Tree Search. The search ends when a complete assignment is found of which distance from the desired win ratio is consistently less than a given termination threshold. If no such assignment is found, the search returns the best complete assignment explored. At each step, the search tries to find an optimal value for a variable. It does so by

selecting the next unassigned variable and randomly selecting a set of possible values for it. The nodes corresponding to values that are not selected are labeled with a number slightly larger than the termination threshold. The two optimizers, which run within the same phase, exchange their candidate values to create a combined partial assignment. Each such partial assignment is extended to a complete assignment by randomly choosing possible values for the remaining variables. In phase one, the remaining features of the board (colors and lengths) are randomly selected to produce a full game board. In phase two, the complete assignment represents a full game board. Twenty games are played on the full game board, and then the distance between the average winning ratio of the played games and the desired winning ratio is calculated. The optimizers return the complete assignment if the distance is below the termination threshold. If the distance is above the termination threshold, each optimizer labels the node corresponding to the current value with the distance and moves to the next value. After trying all selected values without returning a complete assignment, the optimizer chooses the lowest labeled node from the current and previous steps and continues the search. By labeling the unselected values with a number slightly larger than the termination threshold, we ensure that as long as the selected values are labeled with numbers larger than the termination threshold, new values will be chosen.

The algorithm is guaranteed to terminate as it either finds an acceptable complete assignment or it exhausts the search space. There is a theoretical chance that the label of a node is between the termination threshold and the label for unselected nodes. In this case, the algorithm may cycle forever. We did not encounter such a case in our experiments.

The Destination Value Optimizer (DO)

The Destination Optimizer is provided with a set of candidate destinations and a set of possible values to be assigned to the destinations. The special value of -1 denotes that the destination it was assigned to will not be part of the final game.

The Single-Double Route Optimizer (SDO)

This optimizer considers each pair of connected cities and decides whether to make the route connecting them double. It tries to find the optimum number of double routes between two extremes: having too few may result in very limited access to an important city, which is undesirable; having too many may remove conflicts over routes, making the game less competitive.

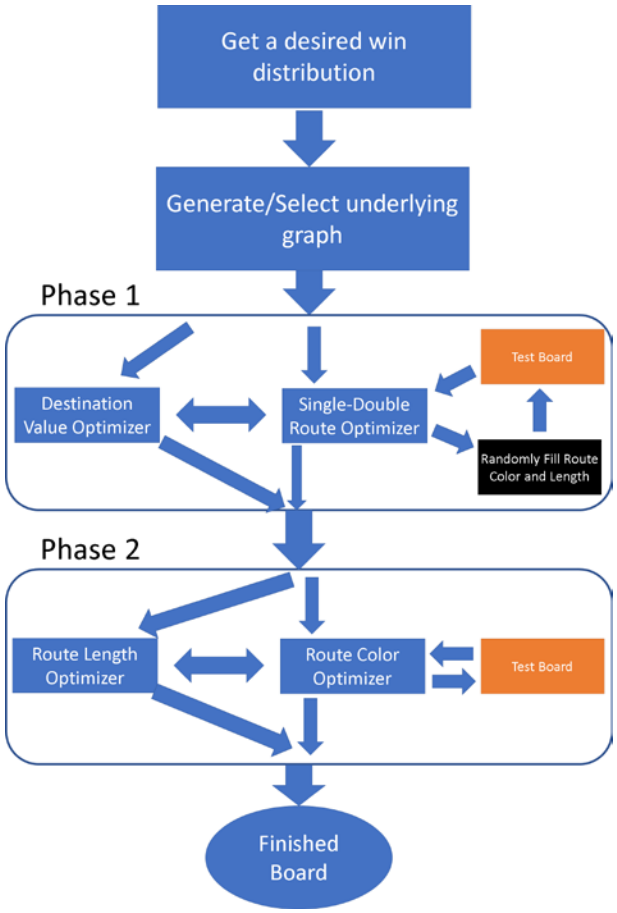

Figure 2. Map building strategy.

The DO and SDO operate synchronously and share information after each choice. To simulate games, they use random route length and route color assignments. Since the random assignment of features produces a high variance of the game results, the termination threshold for these simulations was set to 0.2.

All the routes and destination cards are set at the end of phase one. Phase two proceeds to select the route lengths and colors. As per the requirements of TTR, double routes will be assigned the same length.

The Route Length Optimizer (LO) and the Route Color Optimizer (CO)

For each step, these optimizers select the best four choices of their respective feature to further explore. All the combinations of these choices, 16 in total, are evaluated through random simulation. The combination with the best average score is selected, and the corresponding length and color respectively are chosen by the optimizers.

Experiments and Results

We run three types of experiments:

- Generate a new game board on the underlying graph of the original TTR. In this case, we start with the underlying graph of the original USA board, and optimize the destination values, the double routes, the route lengths, and the route colors.
- Generate a completely new game board. In this case, we apply the full map building strategy as described in the previous section.
- Generate new game boards by connecting 3 subgames and then optimizing. This is a variation of the previous case in which 3 independent subgames (corresponding to subgraphs produced by GO) are generated. These subgames are then connected and partially optimized to create the final game.

Notice that the last two experiments consist of generating completely new games.

To produce a reliable win rate, we simulate 1000 games on the generated game boards.

For each experiment, we compare the results with those obtained from entirely random game boards, which are produced by randomly selecting all the features added to the underlying graph. We estimate the distance of the random game boards by running 1000 simulations on 10 randomly generated boards on the same underlying graph.

Game Boards Generated by Optimizing The Original Game Features

We run a complete and a restricted version of optimization on the underlying graph of the original TTR game, USA board.

The complete version optimizes all the features without restrictions. In the restricted one we impose an extra constraint on the length of a route: it can deviate at most one unit from the length of the original game. The reason for this restriction is to preserve some of the geographical distances between cities. It will be hilarious to have a map on which the distance between Dallas and Houston is six units while the distance between Los Angeles and El Paso is one.

The results presented in Tables 1 and 2 show that both types of optimization result in improvements over the original and randomly produced maps. The HA is favored in all cases. This seems to be related to the original design of the game, most probably linked to the value of destination cards. Fully optimizing the route lengths allows more room to adjust destination values. Thus this type of optimization produces the best results.

In Figure 3, we present the map produced by the restricted optimization applied to the underlying graph of the original TTR.

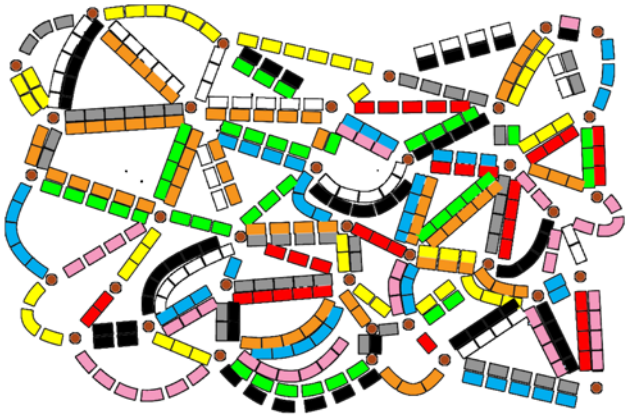

Figure 3. Restricted Optimized Ticket to Ride board.

Agent	Desired	Original Game Board	Complete Optimized Game Board	Restricted Optimized Game Board
OSTA	0.25	0.2025	0.215	0.212
PA	0.25	0.0325	0.151	0.141
LRJA	0.25	0.3045	0.251	0.243
HA	0.25	0.4605	0.382	0.402

Table 1. Comparison of average win ratios for original and optimized game boards generated on the original underlying subgraph.

Original Board	Complete Optimized Board	Restricted Optimized Board	Random Board
0.212	0.1068	0.121	0.1402

Table 2. Comparison of average distances for original and optimized game boards generated on the original underlying subgraph.

Map	Total Routes	Avg. Dest. Card Value	% Double Routes	Total Trains	Avg. Route Length
Orig	100	11.6	28.2	307	3.70
Rest Opt	123	19.2	57.7	417	3.39
Com Opt	115	15.133	47.4	368	3.20

Table 3. Comparison of features of original TTR board and the optimized versions.

In Table 3, we present the comparison of main features in the original and optimized versions. As Table 3 shows, the destination values are increased, and so is the number of double routes. These changes are a disadvantage to the HA

Color	% Routes			% Total trains		
	Orig	Rest Opt.	Com Opt.	Orig	Rest Opt.	Com Opt.
Gray	44	11.4	10.4	30.2	10.0	12.0
White	7	8.9	13.0	8.47	10.5	7.9
Black	7	12.2	12.2	8.79	12.9	12.2
Orange	7	14.6	10.4	8.47	16.1	12.8
Red	7	8.9	8.7	8.79	8.6	6.8
Pink	7	9.7	12.2	8.79	9.8	15.2
Yellow	7	12.2	9.6	8.79	10.8	9.0
Blue	7	11.4	12.2	8.79	10.5	10.9
Green	7	10.6	11.3	8.79	10.5	13.3

Table 4. Comparison of route color distribution of original TTR board and the optimized versions.

Length	Point Value	% Routes Original	% Restricted Optimized	% Complete Optimized
1	1	9	11.4	20.0
2	2	36	16.3	24.3
3	4	20	27.6	13.0
4	7	18	19.5	14.7
5	10	8	17.1	13.9
6	15	9	8.1	13.9

Table 5. Comparison of route length distribution of original TTR board and the restricted optimized versions.

since there are more high value destinations for other agents to claim. Additionally, the HA cannot cut off high value routes since most are double.
Total trains increase, but the average route length decreases. This modification limits the LRJA advantage.

In the original game, the total length of routes of each color but gray is the same. On the optimized board, routes of different colors have different distributions; for example, orange routes cover 16% of the total length, while red ones cover 8%. This creates an imbalance between the train cards, which will make some very desirable while others will be mostly avoided.

In the optimized game boards, there are many same color adjacent routes. This forces players to compete for these colors. We believe that this is an immediate result of the increase in the number of double routes, which reduces the competition on routes. In the optimized versions, the competition is veered towards cards. This gives equal opportunities to each player no matter what strategy they use.

In the original board, short routes are prevalent, and the most common ones are of length 2. The optimized one has a more balanced distribution of route lengths, but still favors the shorter ones. Longer routes are more difficult to build but are worth more points. We believe that the increase in the routes' length forces players to choose longer routes and thus diminishes the advantage the LRJA and the HA have on the original game.

Agent	Desired	Optimized
OSTA	0.25	0.3079
PA	0.25	0.123
LRJA	0.25	0.2907
HA	0.25	0.277

Table 6. Average win ratios for a game board with a generated Underlying Graph.

Optimized Board	Random Board
0.10103	0.185

Table 7. Comparison of average distance for random and optimized game boards with a generated Underlying Graph.

Agent	Desired	Optimized
OSTA	0.25	0.241
PA	0.25	0.091
LRJA	0.25	0.29
HA	0.25	0.378

Table 8. Average win ratios for a game board generated from 3 subgames.

Optimized Board	Random Board
0.134	0.178

Table 9. Average distances for a game boards generated from subgames.

Game Boards with Generated Underlying Graph

For this experiment, we used the Graph Optimizer to produce a new underlying graph. We then generate an optimized game board and 10 random game boards. On each board, we play 1000 games. While this represents only one experiment, we believe that it indicates that our approach produces balanced games. Running a single experiment requires more than 48 hours, and this is the only reason for the scarcity of data points.

As shown in Tables 6 and 7, this game board is more balanced than those generated on the original underlying graph. We attribute these improvements to the performance of the Graph Optimizer, which aims to produce underlying graphs which are better suited for creating balanced board games. However, the PA is still far from having an even win rate.
double routes, lengths, and colors.

Game Boards Generated by Combining Subgames

In this experiment, we start by optimizing the three subgraphs chosen by GO, thus producing three independent sub-games. We then connect the underlying graphs of these games in the same way in which GO does

it. The newly added edges are optimized for double routes, lengths, and colors. We run DO to optimize the destinations over the entire graph. Finally, we run a restricted LO, similar to what we did on the original TTR graph, to balance the route lengths. This step is required to create "geographically" consistent game boards. The results (presented in Tables 8 and 9) show an improvement over the randomly generated game boards. However, they are worse than those obtained in the previous experiment. We attribute this to the lack of global optimization for

Interpretation of Results

The results produced by our method are better than their random counterparts. The distance between desired and computed win ratios is low for optimized boards. In most cases, the HA is still favored, likely due to the wide availability of routes and combinations of destinations it can exploit. Also, due to the PA's inability to take more destinations, we expect to see it typically perform worse than the other agents. This does, however, confirm that destinations play a strong role in the likelihood that a player wins. Even when there are many routes to use, the main component of gameplay, building destinations, strongly influences the game outcome.

We discovered some interesting patterns in the produced maps. Longer routes are generated, most likely to force all players to use them, thus reducing the advantage LRJA has from focusing on them. The maps tend to have several adjacent routes of the same color. This creates increased competition on these colors among players whose destination cards require passing through a city that connects these same color routes. Thus, a feature of gameplay, not shuffling in train car cards until all are exhausted, becomes a forefront issue in the game. It does not arise in the original game due to the large number of gray colored routes.

Limitations

Our inquiry assumes that a player follows the same strategy throughout the entire duration of the game. This may be unrealistic for human players who may follow a certain strategy for most of the game and change it towards the end. For example, a player may follow the HA strategy for most of the game and, towards the end, switch to the strategy of the LRJA.

The original game rules allow the use of double routes only when there are at least four players. Since we experimented with four agents, we implicitly assumed that double routes were present. Thus, our method, in its current form, does not extend to a game with two or three players.

Improvements and Future Work

Despite aiming for an equal winning ratio among all agents, all optimized boards favored the HA. In future inquires, it may be interesting to impose a reduced winning rate for this agent, hoping that it will result in a more balanced game.

The boards produced by our method deviate substantially from the original game in terms of the number of routes and distribution of colors. This issue can be rectified by imposing extra constraints on optimizations, such as an upper bound on the total number of routes combined with lower bounds on the number of routes of each color.

In phase one of the proposed method, the colors and lengths of routes – required to produce a full game board- are randomly chosen. A better understanding of the influence of these factors on the winning ratio may produce a heuristic for selecting them. Such a heuristic can improve the estimation of the distance and reduce the number of simulations, which will decrease the running time of the method.

Conclusions

In this paper, we propose a method for generating new game boards for the Ticket to Ride Game, which allows selected player strategies to have more uniform chances of winning the game. The method consists of several phases in which different features of the game are optimized. We employed four different player strategies. Our experiments indicate that the new method produces games that do not favor or penalize any of the studied strategies. We interpret the modifications made by our method to the original TTR game and indicate how they help create a more even competing field for the studied agents.

References

Abukhait, J.; Aljaafreh, A.; and Al-Oudat, N. 2019. A Multi-agent Design of a Computer Player for Nine Men's Morris Board Game using Deep Reinforcement Learning. *In Proceedings of the 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*.

Brinkmann, G. and McKay, B. 2007. Fast generation of planar graphs. *Match-communications in Mathematical and in Computer Chemistry* 58(2).

Coulom, R. 2007. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *In Proceedings of the 5th International Conference on Computers and Games* .

Silva, F.; Lee, S.; Togelius, J.; and Nealen, A. 2017. AI-based Playtesting of Contemporary Board Games. *In Proceedings of the 12th International Conference on the Foundation of Digital Games.*

Silva, F.; Lee, S.; Togelius, J.; and Nealen, A. 2018. Evolving Maps and Decks for Ticket to Ride. *In Proceedings of the 13th International Conference on the Foundations of Digital Games.*

Schadd, M. and Winands, M. 2011. Best Reply Search for Multiplayer Games. *IEEE Transactions on Computational Intelligence and AI in Games 3(1).*
Witter, R. and Lyford, A. 2020. Applications of Graph Theory and Probability in the Board Game Ticket to Ride. *International Conference on the Foundations of Digital Games.*