OSM Ticket to Ride

Wenzel Friedsam
University of Innsbruck
Innsbruck, Austria
wenzel.friedsam@student.uibk.ac.at

Tobias Rupp
University of Stuttgart
Stuttgart, Germany
rupp@fmi.uni-stuttgart.de

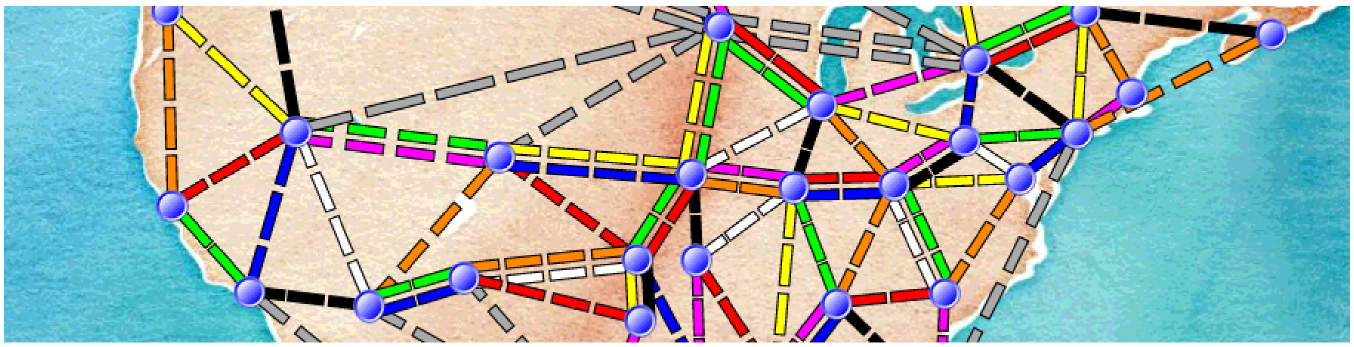


Figure 1: Resulting game graph of the United States using our algorithm.

ABSTRACT

Board games such as *Ticket to Ride* by Days of Wonder use game boards which are based on real geographic data. In this particular game, the map represents an abstract railway graph. Generating such maps by hand is a time consuming process. Therefore we present an algorithm that can generate maps for *Ticket to Ride* using OpenStreetMap data as input. The result can be imported in an open source web version of *Ticket to Ride*, which allows players to play on the generated map.

CCS CONCEPTS

• **Applied computing** → **Cartography**; **Computer games**.

KEYWORDS

OpenStreetMap, railway network, map generation, board games

ACM Reference Format:
Wenzel Friedsam and Tobias Rupp. 2024. OSM Ticket to Ride. In *17th ACM SIGSPATIAL International Workshop on Computational Transportation Science GenAI and Smart Mobility Session (IWCTS'24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3681772.3698211

1 INTRODUCTION

Board games often feature abstract maps which are based on real geographic data. One example is *Ticket to Ride* (TTR), a board game for two up to five players designed by Alan R. Moon and published by Days of Wonder in 2004. The game is quite popular with several million copies sold and won various awards including the German "Spiel des Jahres" award. The map represents an abstract railway graph of North America, containing a set of train stations which are connected by railway routes. The goal of the game is to earn points by placing train cars on these routes to complete destination card tasks. Each destination card specifies two cities on the map that must to be connected through built routes. In general, there are several possible paths to achieve this, but not all are equal efficient. The challenge of the game is to build the required routes in time before they are occupied by other players.

Generally, board game maps are planned and created by hand, which can take a lot of time. At the present time, where many tasks can be automated with current hardware and software, the question arises whether the creation of such playable maps can also be automated. We created an application that can generate new TTR maps based on the freely available geodata from *OpenStreetMap* (OSM) [1].

In the following, we describe the original game and its existing open source version. Then we formalize our quality criteria for the map generation and outline our implementation.

1.1 Related Work

There are already existing projects using OSM data to generate game maps for board games. One example [5] is a tool to automatize the map generation process for the game RISK by Hasbro. Geographic administrative boundaries from OSM are extracted, compressed and processed to generate a map for the game, a collection of territories. To make it playable, a neighborship graph representing the adjacencies of territories and a hierarchy to form groups of territories are generated.

In [4], new maps for TTR are generated based on geographical data in so far that they pick real world cities as city nodes for their boards by hand. However, their connections between cities are rather unrealistic since they are generated by a genetic algorithm not taking real geographical data into account.

IWCTS'24, October 29-November 1, 2024, Atlanta, GA, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1151-0/24/10.
https://doi.org/10.1145/3681772.3698211

In [7], artificial intelligence is used to generate TTR maps. Their focus is to make the game more enjoyable by evening out winning chances. They modify a random graph generated using a set of rules or the original TTR board by changing certain parameters like the length of routes. While this approach is also an example for automatically generated TTR maps, the graph is not based on any geographic data.

In our approach we focus on generating maps based on the real railway network while keeping the graph as similar as possible to the original game graph in terms of relevant game metrics.

2 TICKET TO RIDE

There are different variants of the game with slightly adapted content and rules, for the sake of simplicity we only refer to the original version and its game rules [6] in this paper.

The game map consists of 36 different cities which are connected by routes (edges). Each route has a defined length in segments and a color. Some routes are double-routes and can have two different colors. Each route can be claimed by one player. Train car cards can be spent to claim a route. At the beginning of the game, each player receives three destination tickets. Each destination ticket lists two cities, the goal is to establish a connection between these cities.

The game is round based, in each turn a player can perform exactly one of these three actions:

(1) **Draw train car cards** from the deck or a set of five face up cards. There are eight different card colors and a locomotive card (joker). When a locomotive is drawn from the face up cards only one card can be drawn, otherwise two.
(2) **Claim one route:** To claim a route one train car card per route segment is required. All cards need to match the color of the route on the map. If the route is gray, any color can be used, but all cards still need to have the same color. Locomotives are joker and represent every color. After claiming the route, the player places one train car per segment on the board map.
(3) **Draw three new destination tickets.** At least one needs to be kept.

The goal of the game is to reach the most points by claiming routes and completing destination tickets. In addition to that you can get bonus points by having the longest continuous path.

2.1 Ticket to Ride Web Version

There is an open source web version of TTR created by Kiryushin Andrey [3]. The default map is a railway graph of Russia, but there is also an option to import `.map` files to play on custom maps. Our implementation outputs a compatible file that can be imported there.

3 GAME GRAPH GENERATION

We use OpenStreetMap as our data source in order to extract train stations and the railway network. Our goal is to generate a game graph and export it as `.map` file that can be imported in the open source web version of TTR. To achieve this, we need a list of train stations with a defined latitude and longitude and a list of undirected edges connecting these stations. Each edge has a defined length and width in segments. To simplify the process, the generation is split into several algorithms. The code of our implementation can be found at [2].

3.1 Quality Criteria

To ensure the generated game graph is suited as TTR map, we define a list of criteria that must be followed during the generation process.

Q.1 **Graph is connected:** If there are multiple connected components, the largest connected component is chosen.
Q.2 **Amount of stations, edges and segments:** In the original game, there are 36 different stations, 78 different edges and 307 total segments. To keep the game graph as similar as possible to the original map (in terms of basic graph properties), it should have the same or very similar amount of stations, routes and segments.
Q.3 **Station position distribution:** Selected Stations should be distributed evenly over the game map. Otherwise some parts of the board might be empty and others crowded with stations. In addition to that each station needs to have a sufficient distance to all other stations on the map.
Q.4 **No edge intersections:** The graph must be planar and no edges must intersect.
Q.5 **Minimal angle between edges:** Adjacent edges must have a minimum angle that results in a sufficient distance between both edges.
Q.6 **Edge properties:** Each edge has a length of 1 to 6 segments and a width of 1 or 2 segments, as in the original game.
Q.7 **Segment length:** All individual segments must have a similar length on the board map.
Q.8 **Unique station names:** Names are used to identify each station and therefore must be unique.

3.2 Station Selection

Depending on the input data, there are up to several thousand train stations that get extracted from the OSM file. For the final game graph, we select the most relevant stations that are evenly distributed over the map. For that, we define a station score $f(s)$ to capture the relevance of each train station by factoring in the amount of rails of a station and the number of direct neighbors. Then we selects train stations greedily so that the station score and distance to already selected stations is maximized.

We implement a station selection algorithm with 36 iterations, one for every station to be added. The input is a set of all stations S_{All}, the output a set of selected stations $S_{Selected}$, $|S_{Selected}| \leq 36$. In the first iteration, we select the train station s_{max} with the highest station score $f(s)$:

$$S_{Selected} \coloneqq \{s_{max}\},\ s_{max} = \underset{s \in S_{All}}{\arg\max}\, f(s)$$

For all following iterations, we select the station where the score multiplied by a distance factor $disf$ is maximized.

$$S_{Selected} \coloneqq S_{Selected} \cup \{s_{max}\},\ s_{max} = \underset{s \in S_{All} \setminus S_{Selected}}{\arg\max}\, f(s) \cdot \mathrm{disf}(s)$$

Before we are able to calculate this factor, we determine for each station $s \in S_{All}$ the minimum distance d_{min} to all already selected stations $s' \in S_{Selected}$. The function $distance$ returns the air-line distance between two stations.

$$d_{min}(s) = \min_{s' \in S_{Selected}} distance(s, s')$$

Then we can calculate the distance factor which is normalized by dividing through the largest minimum distance:

$$disf(s) = \frac{d_{min}(s)}{\max_{s^* \in S_{All} \setminus S_{Selected}} d_{min}(s^*)}$$

A simple example of the station selection process is shown in fig. 2.

At the moment the distance factor has a linear and fixed influence on the station score. In our implementation we added an exponent e to modulate its effect. The higher the exponent, the faster does $\text{disf}(s)^e$ decrease for smaller minimum distances and the more important becomes the equal distribution of the stations as a result, we use $e = 2$ as default.

3.3 Edge Selection

The next step is to choose a suited set of edges that connect the selected stations. We start by extracting all railways from the OSM file and then create a list of all possible routes between the selected stations. Afterwards, we choose edges using a greedy algorithm that calculates a score for every edge and selects the edge with the lowest score until the target amount of edges is reached. The score is calculated using three multipliers:

(1) The edge length: Shorter edges are selected first.
(2) The maximum perpendicular distance between the original railway and the straight edge: Smaller distance, i.e less error is better. An example is shown in Figure 3.
(3) The maximum angle to neighboring edges which are already selected: This results in more spread out edges from the station nodes.

Edges that intersect with already selected edges are discarded.

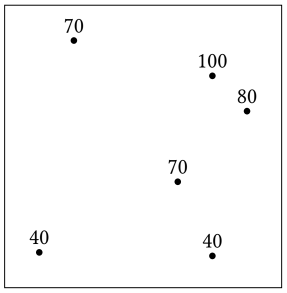
3.4 Edge Properties

The final step in the generation process is adding properties to each edge. As defined in section 3.1 Q.6, each edge has a length of one to six segments and a width of one or two segments (double-route). Our goal is to define edges which have a high chance of being required for a shortest path between two train stations as double routes. As a result, these edges can be occupied by two players, while edges with a width of one segment can only be used by one player.

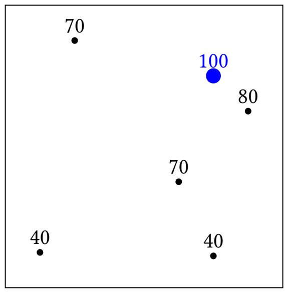
To achieve this, we calculate the shortest paths between all stations. Edges that are part in a lot of these paths are marked as double-routes. The length in segments is determined to be roughly linear to the length of the straight edge on the map. Small modifications ensure that the desired amount of segments on the map is reached.

3.5 Results

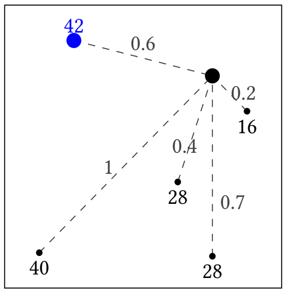
Fig. 4 shows the result when using the OSM data of Germany as input. In general, the results can be divided into two groups: Datasets with a dense or sparse railway graph as input for the algorithm.



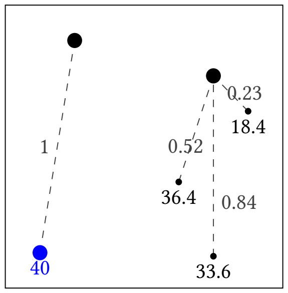
(a) In this example our goal is to select four out of six train stations. The numbers represent the station score.



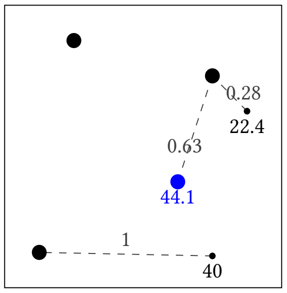
(b) Iteration 1: We select the station with the highest station score $f(s)$.



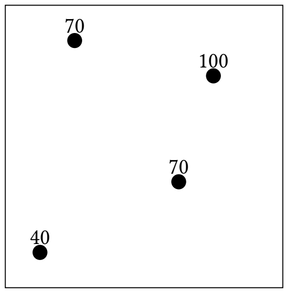
(c) Iteration 2: We select the station where $f(s) \cdot \text{disf}(s)$ is maximized. The dotted lines represent $\text{disf}(s)$.



(d) Iteration 3: Third station is selected.



(e) Iteration 4: Last station is selected.



(f) Final result: A good compromise between importance and distribution.

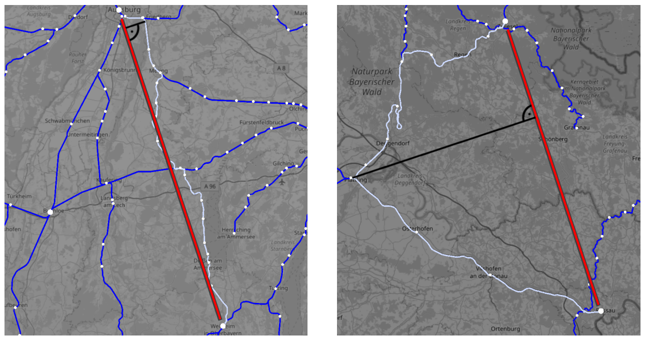
Figure 2: Example of the station selection process. Numbers are rounded.

3.6 Dense Railway Network

A solid and connected passenger railway network is a good foundation for great game graph results. Fortunately, this is the case in most European countries and also other tested countries as India. For such data sets, our algorithm produces a reliable result and is able to reach the edge target of 78 edges.

3.7 Sparse Railway Network

The continent Africa was chosen in order to test the algorithm on a dataset with a sparse railway network. The station graph consists of several different (interrelated) components that are not connected



(a) Relative max. perpend. distance: 0.065

(b) Relative max. perpend. distance: 0.638

Figure 3: Visualization of different perpendicular distances. Small dots represent train stations, large dots are selected stations. Background: ©OpenStreetMap contributors.

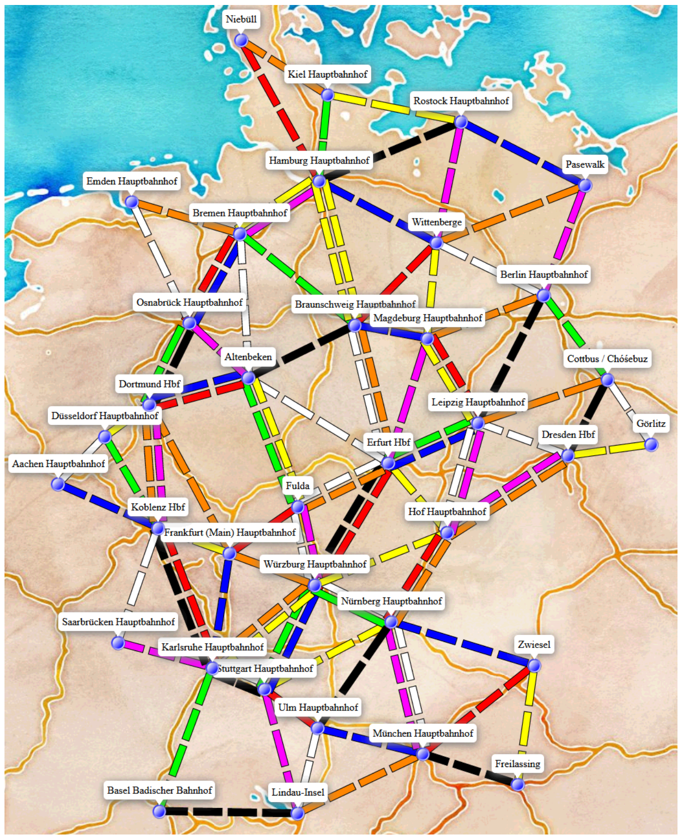


Figure 4: Final result imported in the TTR web version. Map tiles by Stamen Design under CC BY 3.0, Data ©OpenStreetMap contributors.

by each other. In order to generate a connected game graph, only the largest connected component of Africa's railway network is used: The region in and around South Africa. The generated game graph has a total of 51 edges instead of the target of 78. On the one hand this is a general problem for countries with such sparse railway graphs, on the other hand the algorithm is still capable to generate a playable game graph.

4 CONCLUSION AND FUTURE WORK

We created an application capable of automatically generating playable maps for the game *Ticket to Ride* using freely available geographic data from OpenStreetMap. The output can be imported into an open source web version of TTR. By defining a list of quality criteria, we ensure the generated graph is suited as playable game map. In addition to that, our goal is to keep the created game graph similar to the underlying railway network while keeping it as similar as possible to the original game map in terms of relevant game metrics. The algorithm is capable of generating results for countries, continents or the whole planet.

4.1 Future Work

Two steps in the map generation process, coloring edges and the generation of destination tickets, are performed after the import by the open source version of TTR. For future work, an own implementation of these two algorithms could be considered, which might be able to achieve better results, specially adapted for our generated game graph.

Also, we only consider the original version of TTR for our map generation. There are other versions with different game graph characteristics and also different rules that impose new challenges to the creation process.

REFERENCES

[1] 2024. OpenStreetMap Wiki. ttps://wiki.openstreetmap.org/wiki/About_OpenStreetMap.

[2] 2024. OSM-TicketToRide Source Code. https://gitlab.com/CHfCGS/osm-tickettoride.

[3] K. Andrey. 2020. Ticket to Kotlin-building an online board game. https://medium.com/gitconnected/ticket-to-kotlin-building-an-online-board-game-8ac8466fe142.

[4] Fernando de Mesentier Silva, Scott Lee, Julian Togelius, and Andy Nealen. 2018. Evolving maps and decks for ticket to ride. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*. 1–7.

[5] Patrick Lindemann. 2021. OpenStreetMap risk maps. , 44 pages. https://doi.org/10.18419/opus-12011

[6] Alan R. Moon. 2019. Ticket to Ride, The Cross-Country Train Adventure Game. https://ncdn0.daysofwonder.com/tickettoride/en/img/7201-T2R-Rules-EN-2019.pdf

[7] Iain Smith and Calin Anton. 2022. Artificial intelligence approaches to build ticket to ride maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 12844–12851.

accepted 27 September 2024