Simulation of Vienna metro system

Simulation and Performance Evaluation, Final Project

Alessio Blascovich
alessio.blascovich@studenti.unitn.it

Matteo Di Noia
matteo.dinoia@studenti.unitn.it

***Abstract*—The metro infrastructure represents nowadays a crucial public service offered in big cities. These systems are considered space and time efficient, and as such they occupy an increasing major role in public transportations. Managing such a big infrastructure come at a cost, the difficulty to estimate its performances.**

Here we present a simple and self-made Discrete Event Simulation (DES) simulation engine. To test it we performed a study case of the Vienna metro system. In particular, we focused on the impact of trains failures and delays over the service fairness.

I. Introduction

In this paper, we present a DES of the train and passenger flow of the Vienna metro. We decided to analyse this problem, as the current increasing importance of the public services. This is because we are finally seeing the limitations of the current most common way of transportation, cars.

Underground public transportations can increase the maximum number of people moving around up to 20 times [1]. In an historical city such as Vienna, where massive street projects cannot take place, these types of transports are crucial to optimise the ease of movement.

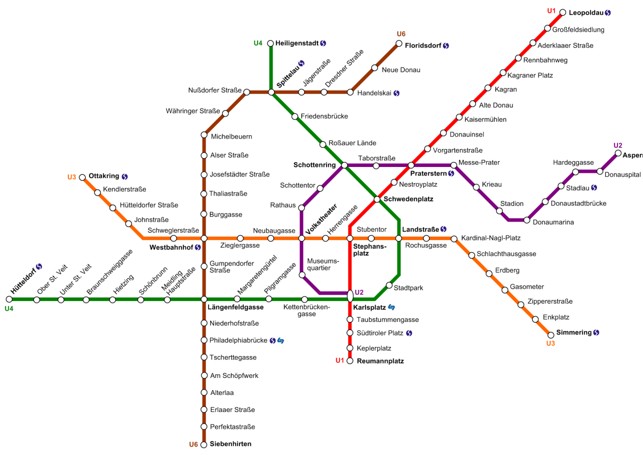

Fig. 1. Map of Vienna metro

II. Related work

The event-driven simulation by Grube et al. [2] used the same programming paradigms we applied. They used an object-oriented approach to represent various entities within the simulation, and an event-based system to keep the simulation going and collect data.

Schmaranzer et al. [3] also simulated the Vienna metro system, using an architecture and an approach similar to ours, but they also keep in consideration time table and train headways.

Some work, like Asgari et al. [4], take into consideration a map with different kind of public transportations. However, due to time limitations our work focuses only on the metro system.

The work of Welch et al. [5] uses the population density to create a better people distribution around the various system entry points.

III. Problem definition / system setup

We decided to separate the simulation from the data analysis. For the simulation engine we used *Rust* since we were both familiar with this language and, the ahead-of-time compilation yields to a much more efficient executable. Also it is optimal for such a phase because of a more robust type system, allowing to write code faster, reducing chance of errors.

The second part of our work focussed on the analysis of data gathered throughout a simulation run. For this task we used *Python* an industry standard language with a huge set of libraries for data analysis; which also allows quick experimentation thanks to its flexibility in data/types handling.

A. Simulation Engine

The engine for our simulation uses a DES paradigm to handle all the events. A Discrete Events Simulation is characterised by a series of instantaneous events, such as a train arrival or departure. Following the DES paradigm the core component of our system is a binary heap that stores all the events in a sorted manner; the system changes its state only upon an event pop from the binary heap. In our engine each event can generate and schedule new events representing the start or end of some action or period of time.

We started from scratch, using external libraries only when strictly necessary. Our simulation engine is open-source and freely available on *Github* [6]. For the map, we used a dataset created and provided by the *OpenMetroMap* organisation [7].

To better model the real scenario as good as we could, we decided to identify and handle the following events:

1) **Start**: an hardcoded events which is used to the bootstrap the simulation and schedule the first real event;

2) **End**: the start of the simulation;
3) **PersonArrive**: represents the arrival of a single person in some station, it also schedules the next PersonArrive;
4) **TimedSnapshot**: a periodic snapshot used to obtain the number of people waiting in the whole system;
5) **TrainDepart**: represent the departure of a train from a particular station, and it is scheduled when handling a TrainArrive, during this event people are also boarded;
6) **TrainArrive**: represent the arrival of a train in a particular station, it is scheduled when handling a TrainDepart;
7) **TrainCrashed**: after a distance given by an exponential distribution a train crashes;
8) **TrainRecovered**: the crashed train gets repaired;
9) **WaitingForEdge**: if a railroad is occupied by another train the current train has to wait in the current station;
10) **FinishedWarmup**: determines the end for the warm-up phase.

Initially the simulation starts with a warm-up phase, we implemented this phase to allow the system to overcome the initial bias before starting collecting data. Since the simulation is quite lightweight we decide to over-estimate the warm-up duration. The warm-up duration is a parameter that can be tuned by simply modify its value within the code and recompiling the software.

The simulation engines uses simple buffered I/O to write on *csv* files various metrics gathered during the simulation. We decided to collect data regarding:

- time required to board on a train;
- delay time to arrive into a station, the delay measured can be negative if the train is ahead of time;
- periodic snapshot of the people present per each line;
- people served between two snapshots.

For the sake of simplicity the simulation engine does some assumption and simplifications:

- trains speed is constant between two nodes and it picked following an normal distribution bounded between two speed values;
- for every station, people arrival is guaranteed by a *Poisson* process;
- at each station, people in a train, randomly decide whether to descend or not (except at line end) and after that they randomly decide to either pick another line (if present at the station) or to leave;
- each directed segment between from a node to another has a capacity that cannot be exceeded;
- train lines cannot fork.

These assumptions and simplifications had consequences on our work. A single person cannot perform a path finding task to a target station, they can only randomly enter and exit the metro system.

Additionally, we cannot determine which station had the higher traffic rate; so we modelled people arrival following a *Poisson* with the same parameters for every station. These two simplifications flattened the traffic making each station rather similar.

In a real world scenario two metro trains cannot overtake each other while on the same railroad with the same direction. Instead of implementing a complex procedure to verify that no overtake take place, we simply impose that in a railroad between two stations there can be at most one metro train per direction.

B. Data Analysis

In order to process the *csv* file produced by the simulation we used different *Python* scripts. Every script parsed and analysed a different metric. We started doing a quantitative review by analysing and plotting the data gathered throughout the whole simulation.

Once completed this early analysis, we proceeded to infer some more complex index, such as the *Lorenz* gap, and *Gini* index.

Initially the data analysed were generated by a simulation engine set using some "default" parameters:

Parameter	**Value**
Train per line	10 [8]
Train capacity	800 [3]
Railroad capacity	1
Space before crash	$\sim \mathcal{E}(1e4)$ km
Restoration time	$\sim \mathcal{E}(10)$ min
People arrival time per station	$\sim \mathcal{E}(20)$ sec
Train speed	$\sim \mathcal{N}(32.5, 20)$ km/s [8]
Train speed interval	[10, 80] km/s [8]
Simulation length	30 days
Simulation warm-up	30 days

TABLE I
SYSTEM "DEFAULT" PARAMETERS

We iterated the analysis for multiple simulation configurations. For each run we modified some crucial aspects of the metro system such as the train crash rate, speed, the number of trains available for each line, *etc*.

C. Data Format

```
stations: [
    {   "lat": real,
        "lon": real,
        "name": string,
    }
]

lines: [
    {   "name": string,
        "color": string,
        "stops": [integer],
    }
]
```

Listing 1. JSON schema used

Finally, in relation to the metro's map format, we noticed that there is not a standard to define them. As such starting from [7] we created a simple JSON, formatted according to Listing 1, which is the input of the simulation. The format picked is quite simple so any map can be easily converted with minimal work to a format understandable to the simulation.

The major current limitation of this approach is that lines cannot have splits in them. As shown in Figure 2, a splitting line must be divided in two or more not splitting lines (possibly partially overlapping).

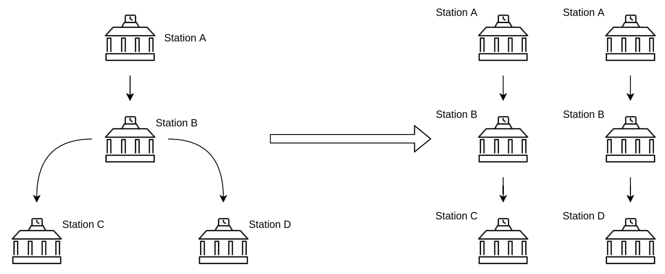

Fig. 2. How a splitting line is converted

IV. Results

For the initial runs we decided tu use "default" parameters for the system. We proceeded gathering basic pieces of information about our system.

A. People served

The first metrics that we decided to measure was the amount of people served. By *served* we categorise all the people that took at least one metro and then left the system. In order to retrieve this data we counted the number of people leaving the system over 10 minutes period. As Figure 3 shows the amount of people is distributed around the empirical mean of 5'880. Following this mean we deduced that our system serves, on average, a total of 35'300 person per hour, so 847'000 person daily. These numbers align quite well with real data-points presented in [8].

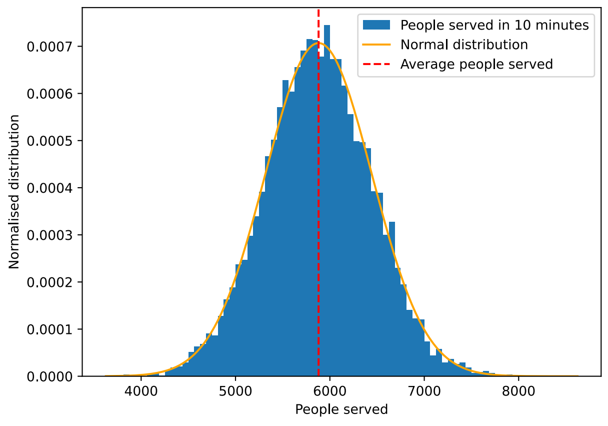

Fig. 3. Distribution of people moved around

Figure 4 pictures more in details the evolution over time of the number of people served by our system. We also compute the moving average which shows that there is a constant trend in the data around the mean. To assure the absence of a trend we use the "lest square" method, to be precise we used the method based on the *Vandermonde*'s matrix. In Figure 5 are summarised the results of this test, and as it is clearly visible there is no trend in the data gathered so far.

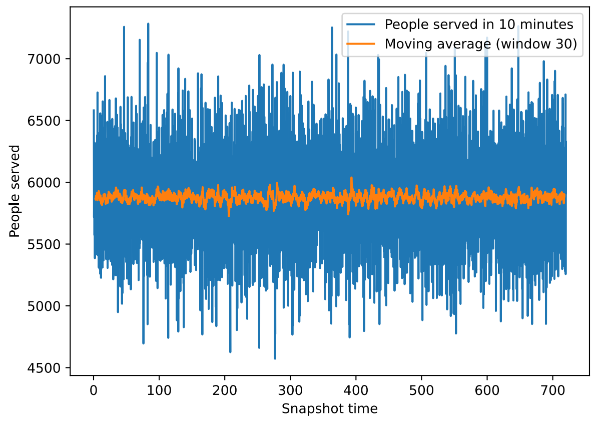

Fig. 4. Evolution of the transported people number

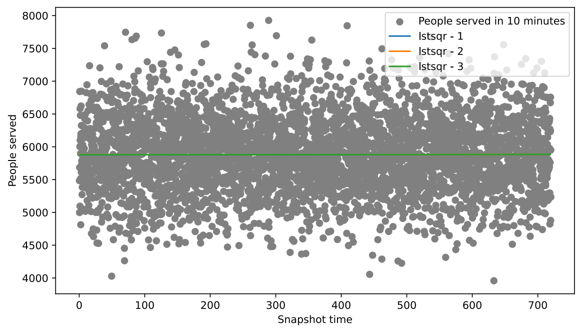

Fig. 5. Scatter-plot of people transported

As testified by the previous test the data are already detrended, so it is possible to use the "Maximum Likelihood parameter Estimation" (MLE) to estimate the parameters generating the data in Figure 4. Since the data are drawn from a set natural numbers and the data are clustered around a single value, we decided that a *Gaussian* could have been a perfect suite for this dataset. The likelihood function is:

$$L(\mu,\sigma^2) = \prod_{i=1}^{n} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2} \right) \right) \quad (1)$$

$$= \frac{1}{\sqrt{(2\pi\sigma^2)^n}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i-\mu)^2 \right) \quad (2)$$

The following steps are computing the log like function of Equation (2); defined as $l(\mu,\sigma^2) = \ln\left(L(\mu,\sigma^2)\right)$. Then to find the estimators $\widehat{\mu}_n$ and $\widehat{\sigma^2}_n$ we need to compute the argmax in μ and σ^2 respectively.

$$\frac{\partial}{\partial \mu} l(\mu, \sigma^2) = 0 \implies \widehat{\mu}_n = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3}$$

$$\frac{\partial}{\partial \sigma^2} l(\mu, \sigma^2) = 0 \implies \widehat{\sigma^2}_n = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2 \tag{4}$$

Equations (3) and (4) represent the MLE formulae for a *Gaussian* distribution. Once obtain those formulae we used the dataset to estimate their values and obtain a suitable *Gaussian* distribution, in Figure 3 we drew the obtained distribution.

B. People waiting

With *waiting* we define the set of people who are waiting for a train to arrive at their station, henceforth the amount of people waiting is the cardinality of such set.

To show that the system is capable of managing the amount of people requesting the service, we plotted the amount of people currently waiting for a train of a specific line. Here we will show only the charts for the evolution of the entire system.

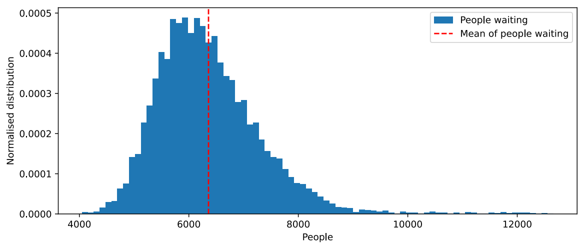

Fig. 6. Distribution of people waiting in the system

As stated by picture Figure 6 the average number of people waiting are 6360, which mean that about 60 people are waiting for each station on average. Also the amount of people waiting is seldom above 8000 and even less frequently above 12000. This could indicates that the system manages to handle the amount of people requesting the service, but to be certain we also plotted the evolution of this number over time (without warm-up period), as it is possible to see in Figure 7.

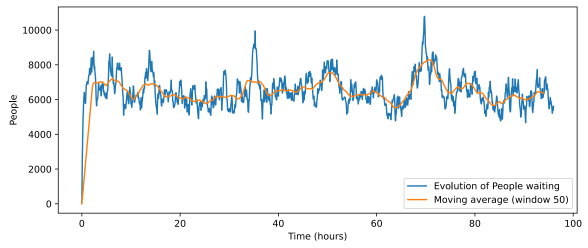

Fig. 7. Evolution over time of the people waiting at a station

From Figure 7 we can evince the fact that the system reach a state of equilibrium rather rapidly. Since we believe the number of people to not be divergent, we expect to have no trend or a stable trend in the data (truncating the warm-up period); to test it we applied the same method we applied before in Section IV-A.

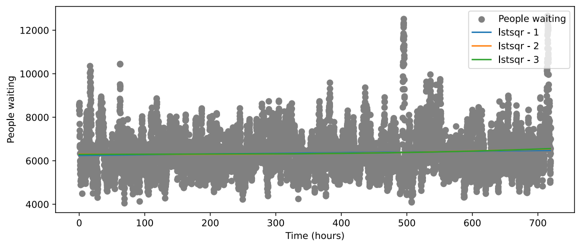

Fig. 8. Scatter-plot of people waiting to get into a train

In the first three iterations of the "Lest Square" method; as plotted in Figure 8 we did not notice any particular skew between the various function produced. Since there is no significant curvature we concluded that the data were trendless, as such the number of people is not divergent.

C. Time to board & delays

To measure the service quality in the current simulation we decided to define and collect the time, on average, a person spent waiting for the next train to arrive.

The data collected are plotted in Figure 9, but the graph is truncate at 60 for readability since the real tail extends up to 100. We did so because the meaningful percentiles lay way lower than 100. Figure 9 shows that, on average, a person spend roughly 10 minutes waiting for the next train; while most of the people (99% of them) wait less than 36 minutes. But this distribution has quite a long tail, i.e., there are some people waiting for a long time; however the tail is values are unlikely. In fact, only a person in a thousands waits for more than 56 minutes.

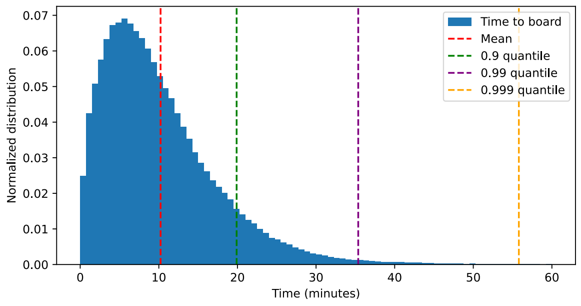

Fig. 9. Time to board approximated distribution

Another quality of service metric is represented by the delays that a train developed during its journey. A delay is every temporal variation on the average travel time between two stations, hence a delay can be either negative or positive. The former means that a train arrived in a station before the

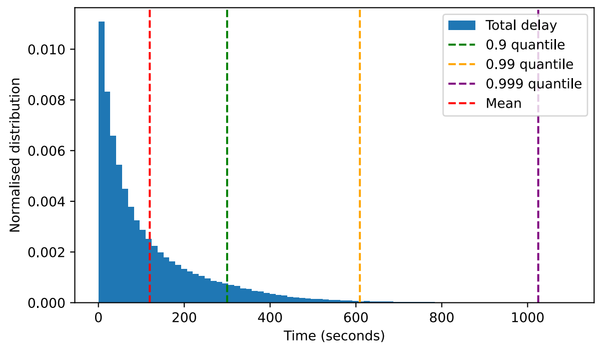

Fig. 10. Delay approximated distribution

estimated time, we did not consider this to be a real issue and so we focused our analysis on the opposite case.

In Figure 10 we reported a partial charts since due its long tail it was difficult to picture. Computing the quantiles we decided to cap the x axe to 1’100 since it covers most of the cases. Quantiles computation also revealed that on average people wait at most 120 seconds for the next train arrival, and 99% of them at most 620 seconds, i.e., 10 minutes and 20 seconds.

D. Fairness indices

To dig a bit more into details about the fairness of our system we used a strategy often used in economics, the *Gini* coefficient; in other words we measured the inequality within the system. The exact value for the *Gini* coefficient could have been computed following Equation (5).

$$\text{MD} = \frac{1}{n(n-1)} \sum_{i,j} |x_i - x_j| \qquad \text{Gini} = \frac{\text{MD}}{2m} \tag{5}$$

However, Equation (5) would have been extremely hard to compute since it requires $O(n^2)$ array scans in order to extract the value of MD. We then opted for an approximation of it that exploits the *Lorenz* gap.

$$\text{Gini} \approx \text{gap} \cdot (1.5 - (0.5 \cdot \text{gap})) \tag{6}$$

Line	Lorenz	Gini
Whole system	0.398	0.518
U1	0.405	0.526
U2	0.387	0.506
U3	0.372	0.489
U4	0.384	0.503
U6	0.399	0.518

TABLE II
LORENZ GAP AND GINI COEFFICIENT VALUES

For a more comfortable view we plotted the *Lorenz* curve for every line within the system and for the whole system.

As it is possible to see in Figure 11, our system is very unfair. The cause of this could be multiple:

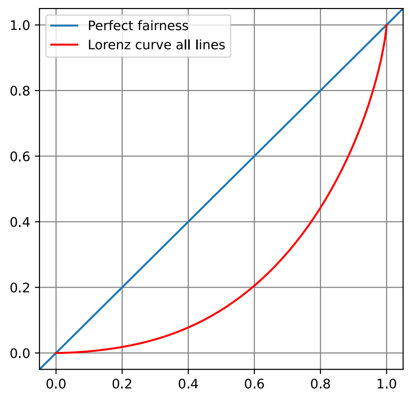

Fig. 11. Lorenz curve for the system

- the absence of a timetable cause people to arrive at random time not necessarily keeping in mind the time at which the train will pass;
- the absence of a timetable make so that there are not connection ready;
- there is the same number of train for each line, even though some line are longer then other, causing some line to have a worse service than other;
- train do not wait if they arrive early.

E. Extreme case of crashing train

A possible usage of our tool could be the estimation of performances, this is useful if the Vienna municipality wants modify its structure. E.g. the municipality is running low on money and has to buy less reliable trains, and fewer squad in charge of repairing broken trains. We supposed that on average a train breaks down every 100km and a squad intervenes in, an average, of 60 minutes.

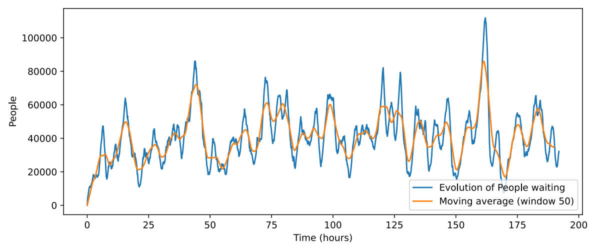

Fig. 12. People waiting in an extreme case

In this scenario (Figure 12) the amount of people within the system does not diverge, but averages around 42’400, with spikes up to 112’000 in some instants. These data suggest that the system is heavily under stress, however the total capacity of train can overcome such load; this explains how some spikes can be handled so rapidly.

In fact, the amount of people served is practically the same, being 5878 instead of 5880; this is expected as in the nondivergent case it is strongly connected to the number of people entering the system.

But while this shows that, from a theoretical point of view, the system is stable, the metrics of the service quality tell another story.

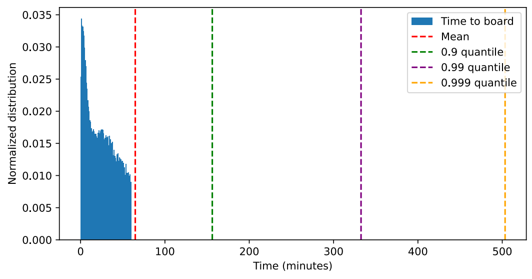

Fig. 13. Time to board distribution in an extreme case

In fact, as it is possible to see in Figure 13, the time to board are strongly impacted by these changes. The average board time is now almost 65 minutes, a value which is strongly skewed by outliers that reach up to 500 minutes, or in other words almost 8 and a half hours. This number are so high that clearly people would have stopped waiting and prematurely exited the system. As such, this data, shows that the system, in the real world would not be able to support the theoretical amount of people from the Figure 12.

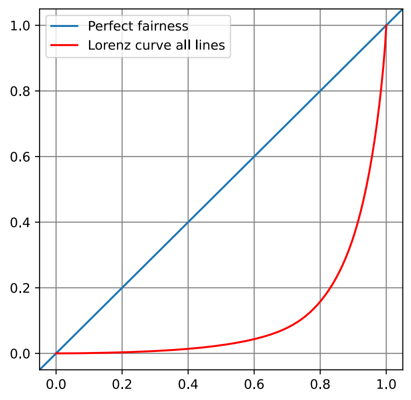

Fig. 14. Lorenz curve in an extreme case

With reference to Table II the *Lorenz* gap almost doubled in every line, the global gap is now 0.644.

This artificially pessimistic study case shown that, the Vienna municipality should not reduce the budget allocated to the public metro system, to such an extreme extent.

V. Future Work

Our simulation is still a bare-bone piece of software, we identified few features that if implemented would greatly amplify the degree of precision of our simulation. We plan to work on them in the following order:

- use of acceleration and deceleration to compute the travel time of a track between two stations;
- use of realistic rate of arrival for people around the station, stations closer to the city centre should be more crowded;
- implementation of the real Vienna metro timetable;
- use of a headway to counteract possible delays;
- have people choose the destination and pick a good path between their position and the destination, instead of having them randomly roam.

VI. Conclusions

Our simulation is still an approximation of the real world, as a matter of fact most of our data aligns with real data [8], such as the number of people served and the time to board; but there are some outliers. For example, the fairness indices are not precise, since our system does not take into consideration real train timetables and people distribution over various stations.

However, this *Lorenz* curve is similar to the one obtained in an adjacent work, Welche et al [5], set in Prince George's County, Maryland, United States.

To conclude, according to the data gathered and explained in Section IV, the Vienna metro system seems to be more than capable to handle its current load. For a public metro being capable to move 800'000 people daily is a strong efficiency indicator especially considering the metro system space footprint.

References

[1] M.-E. Will, T. Munshi, and Y. Cornet, "Measuring road space consumption by transport modes: Toward a standard spatial efficiency assessment method and an application to the development scenarios of rajkot city, india," *Journal of Transport and Land Use*, vol. 13, no. 1, pp. 651–669, 2020. [Online]. Available: https://www.jstor.org/stable/26967263

[2] P. Grube, F. Núñez, and A. Cipriano, "An event-driven simulator for multi-line metro systems and its application to santiago de chile metropolitan rail network," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 393–405, 2011, modeling and Performance Analysis of Networking and Collaborative Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X10001747

[3] D. Schmaranzer, R. Braune, and K. F. Doerner, "A discrete event simulation model of the viennese subway system for decision support and strategic planning," in *2016 Winter Simulation Conference (WSC)*, Dec 2016, pp. 2406–2417. [Online]. Available: https://doi.org/10.1109/WSC.2016.7822280

[4] F. Asgari, A. Sultan, H. Xiong, V. Gauthier, and M. A. El-Yacoubi, "Ct-mapper: Mapping sparse multimodal cellular trajectories using a multilayer transportation network," *Computer Communications*, vol. 95, pp. 69–81, 2016, mobile Traffic Analytics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366416301517

[5] T. F. Welch and S. Mishra, "A measure of equity for public transit connectivity," *Journal of Transport Geography*, vol. 33, pp. 29–41, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0966692313001762

[6] A. Blascovich and M. D. Noia, "Spe-homework," 2025. [Online]. Available: https://github.com/elblasco/SPE-homework/tree/709b2721f036cc41df7df8c80c255b4a90bbbd3

[7] OpenMetroMaps, "vienna," 2019. [Online]. Available: https://github.com/OpenMetroMapsData/vienna/tree/b25aadc1e9c69dd49c50f46e0cc66ba3010e8379

[8] (2022, 05) The vienna metro. [Online]. Available: https://homepage.univie.ac.at/horst.prillinger/ubahn/english/facts.html