AH2173 - Project Report: Itinerary planner based on the previous delays of the vehicles for SL

Atik Ahmed Chodhury, Vianey Darsel

May 27, 2023

The coding of our project can be found on this Github

Abstract

In this project, we focused on addressing the crucial concern of trip planning for public transportation users. The selection of an optimal itinerary holds the potential to save valuable time for these individuals. Typically, users rely on the solution provided by the transport operator. However, our investigation aimed to determine the extent to which we could surpass this solution by incorporating historical network delays into the planner.

To conduct our study, we looked at 6 trips from Storstockholms Lokaltrafik (SL), which provides real-time information that we collected over several days. We analyzed this data to get the real departure times of the vehicles, and so the delay for every journey. We focused, then, on the distribution of the delays, before the departure of a vehicle and during the journey.

Automating route searches based on different criteria has enabled us to create and compare different models. We tested the models for both real-time proposition and 10 minutes in advance proposition. In all cases, the result is that the best-performing model is the one that includes in the construction of the itinerary the median delay.

Contents

1 Introduction

Every day, the Stockholm Public Transportation network carries more than 780.000 [8]. On average, in 2014, a Stockholm citizen makes around 350 trips with SL [1]. That corresponds to one trip per day per resident. The modal split for Public Transport in Stockholm county is 35% [6].

This network is composed of more than 500 lines and a mix of buses, metros, trams, trains, and boats [9]. Since the users cannot know all the network, a major challenge for the organization in charge of running the network is to provide a reliable itinerary planner tool.

To tackle this problem, the more primitive approach is to know the network and to examine the different timetables from all stations. That requires the user to have access to the schedule of all stations wherever and whenever. The flaw of this solution is that, in addition to being humanly difficult to execute, it only takes into account the scheduled journeys and does not take the situation of the network into account. Some disruptions may appear with internal (technical issues, lack of staff..) or external effects (traffic, work on public roads, irregular flow of passengers...). That may cause delays, modifications, or suppression of the service at some stations.

SL has also provided a tool to that extent. The main solution provided by SL is an itinerary planner that is available both on their mobile application and their website [7]. The advantage of this solution is that some of the previous statements are taken into account in the model and it should provide more accurate results. However, some disruptions might have been forecast. Indeed, some delays might appear frequently and can be added to the original trip duration.

In this project, we have tried to integrate this last component into an itinerary planner. For this purpose, we analyzed a couple of itineraries. For each of them, we collected real-time data, observed the delays for each of them, and finally generated a statistically-driven estimation of the best itinerary.

The natural question that this work has tried to answer is whether it is possible to do better than the solution provided by SL by including the history of the delays in the model.

The delays are a major subject of reflection for the operators. Thus, some studies have already tried to tackle this delay issue [5]. However, our approach is different since we are focusing on the user's point of view. Our work is in complement to the operators' solutions. We give a sort of efficiency score of the different itineraries to the users.

2 Scope of the project

We cannot analyze the whole network of Stockholm, so we decided to concentrate on a couple of relevant itineraries. A criterion for the selection of the itinerary was to compare different combinations of modes of transport. Thus, our study is concentrated on 6 trips:

- Tekniska Högskolan from/to Fridhemsplan. This itinerary will compare a direct bus

(Line 4) and a metro with a connection in T-Centralen with another metro (Metro 14 + Metro Blue or Metro Green).

- Tekniska Högskolan from/to Solna centrum.This itinerary will compare a metro with a connection in T-Centralen with another metro (Metro 14 + Metro 11), a bus with a connection in Fridhemsplan with a metro (Bus 4 + Metro 11), and a metro with a connection in Bergshamra with a bus (Metro 14 + Bus 176/177).

- Globen from/to Solna Centrum. This itinerary will compare a direct tram (Line 30) and a metro with a connection in T-Centralen or in Fridhemsplan with another metro (Metro 19 + Metro 11)

To avoid the influence of any external disruption, we have only analyzed the journeys that take place between 7 am and 9 am and between 4 pm and 6 pm during workdays. By choosing the rush hours, we assume that the external parameters are the same at these moments, so there is no influence from the moment when we collect the data in the periods. The second asset of this choice is that it corresponds to peaks in the exploitation of the network, so there are more vehicles in circulation, so we could collect more data.

In the tool that we have created, we have considered three different algorithms:

- a planner that only takes into account real-time data

- a planner that takes into account real-time data and adds a forecast delay based on the median of the delay

- a planner that takes into account real-time data and adds a forecast delay based on the quantile 90% of the delay. That means that the predicted arrival time gives an upper bound which is true at least 90% of the time (based on the previous observations).

3 Method

After collecting the data (Part 3.1), we had to process it for our specific problem to find out the characteristics of the delays for the different journeys (Part 3.2). Then, we created a relevant framework for the analysis of itineraries (Part 3.3.2), which led us to create our own itinerary generator based on different criteria (Part 3.4.3). Finally, we compared this generator with SL one and a timetable-based one (Part 3.5).

3.1 Data collection

3.1.1 Collection of the scheduled planning

To implement a very basic itinerary planner that respects the schedule, we need to have access to the latter. Since this is forecast in advance, we can get this information easily and whenever we want with the second API from SL [2]. The output of this API is a zip that

contains all information needed. There are the details of the shapes of the route of every line, the schedule for each journey and each stop, the planned day of circulation for each journey, the different id names for the different attributes (stop, stations, trip, calendar...), etc...

3.1.2 Collection of SL itinerary planner information

The second piece of information that we want to collect is the real-time proposition from SL. This data is only interesting in our testing stage to compare it with the different models that we implemented. It is collected through the third SL API [3]. We collected trips for the different itineraries that we focused on. To respect the scope of the study, we collected it at the same hours as the analyzed data. The collection was made on the 11th of May afternoon and the 12th of May morning. That corresponds to one complete day for the scope and taking two different days allows us to prevent being biased by any disruption linked to a special day. To have a sufficient data testing set, we ask for each of our 6 itineraries a proposition around every 40 seconds (time may differ due to the other actions in the loop). That corresponds to 2605 successful requests, which would be the baseline for the models' comparison.

From the API, we collected the different itineraries (with the different times, the itinerary, the transfer, etc...). The transfer times for the other models can also be collected by testing different itineraries in this API.

3.1.3 Collection of real-time data

To fulfill our mission, we need to analyze real-time data to get profiles of the delays on the different journeys. The API that is used here is the fourth SL API [4]. This API provides real-time information for the incoming departure of the station put as the argument. We need first to collect the station IDs, which can be found in the API 2[2] and can be stored in a variable for the rest of the data collection. The stations we are inspecting are all departure, transit, and arrival stations. That corresponds to a total of 8 stations: T-Centralen, Tekniska Högskolan, Solna Centrum, Solna Centrum norra, Fridhemsplan, Globen, Bergshamra, and Bergshamra bro. To have the most accurate values, we have made a call to the API every 20 seconds for the training data and 40 for the testing data. The goal here is to have the best estimation and see the evolution of the delays on a line. This amount of calls was possible thanks to an upgrade of the quota of the API from Bronze to Silver, meaning more calls per minute and per month are possible.

Implementation trick:

Sometimes, it happens that the API does not give a response. That implies the code stops running (and no more calls are made). To overcome this issue, we have used the try/expect function from Python, which allows us to go into the except loop when an error was raised in the try loop. Thus, the algorithm will not stop at the first failure.

```
try:
    # treatment of test.json()["ResponseData"]
except:
    # do something when an error should have been raised
    print(station, mode, datetime.now()) #e.g. show parameters
```

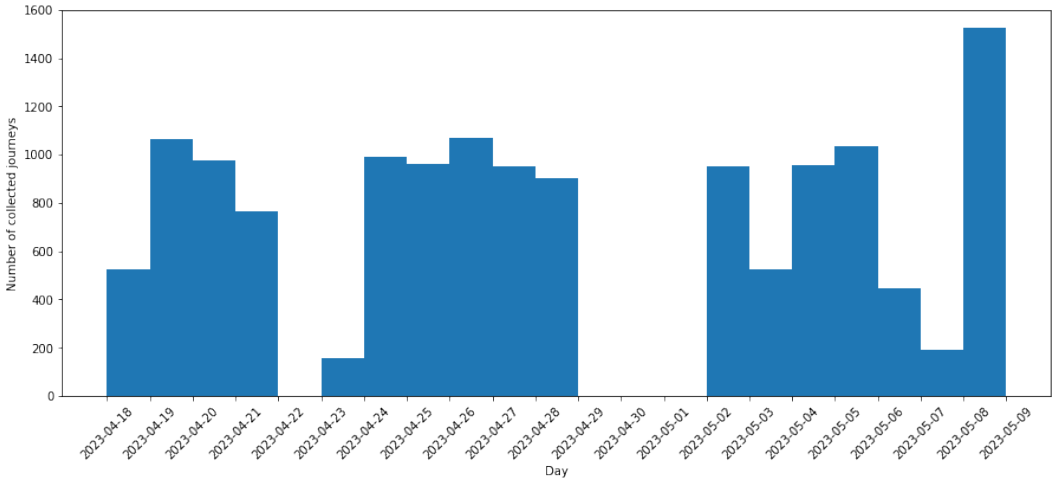


Figure 1: Distribution of collected data over days

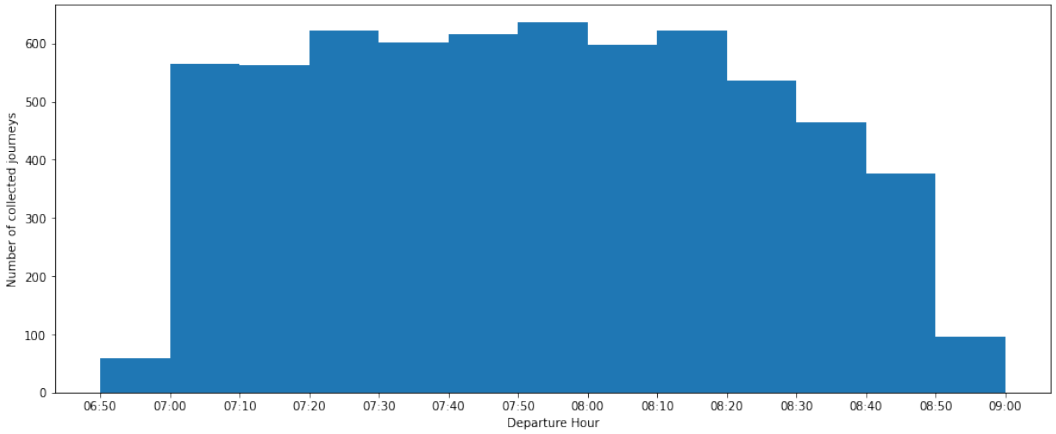


Figure 2: Distribution of collected data over time in the morning

The data is stored in csv files (one per day). No treatment has been made before saving, except adding the time of the call to the API. This procedure is done in order to not lose

any important information in the process. The total amount of journeys analyzed is 13988, dispatched on 18 different days and in two time intervals. The distribution of the collected data is given in Figure 1, 2, 3, 4a and 4b. We are not surprised by the fact that 3 metro lines have more data than the others because there are 3 stations that interested us along Line 14 (T-Centralen, Bergshamra and Tekniska Högskolan), Line 11 (T-Centralen, Fridhemsplan and Solna Centrum) and Line 19 (T-Centralen, Fridhemsplan and Globen). Thus, there is for the same service, three times more possible journeys. Another interesting point from this plot is that there is the number of journeys on Line 4 is the same and even slightly higher than metros' lines. We can deduce that the service is higher on this line is higher during peak hours than on a metro line. Concerning Figure 2 and 3, the extreme values have fewer data because there is no call outside the desired interval and some are at the border. During the different periods, the data is well distributed, which comforts our assumption that the departure time has no influence on the delay. There is a little decrease at the end of each period which is due to the fact that we have plotted here the distribution of the departure time and we have deleted the journeys where the arrival time was out of the defined borders.

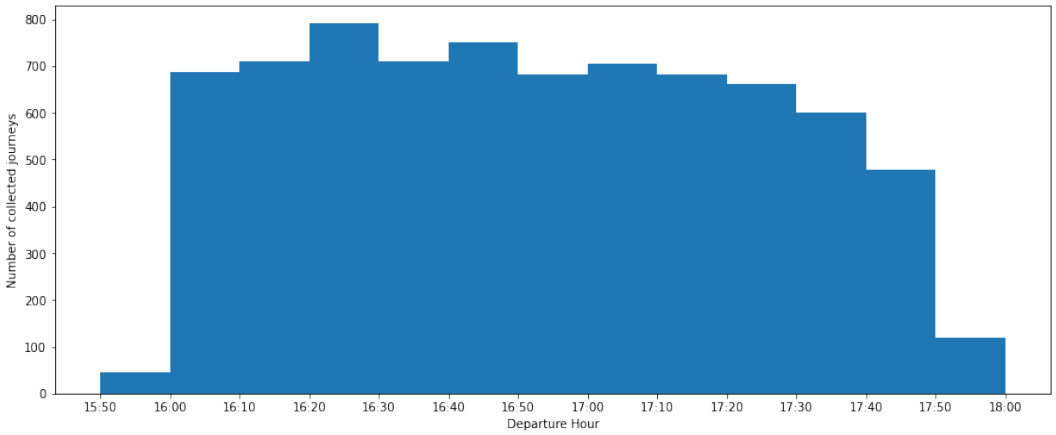
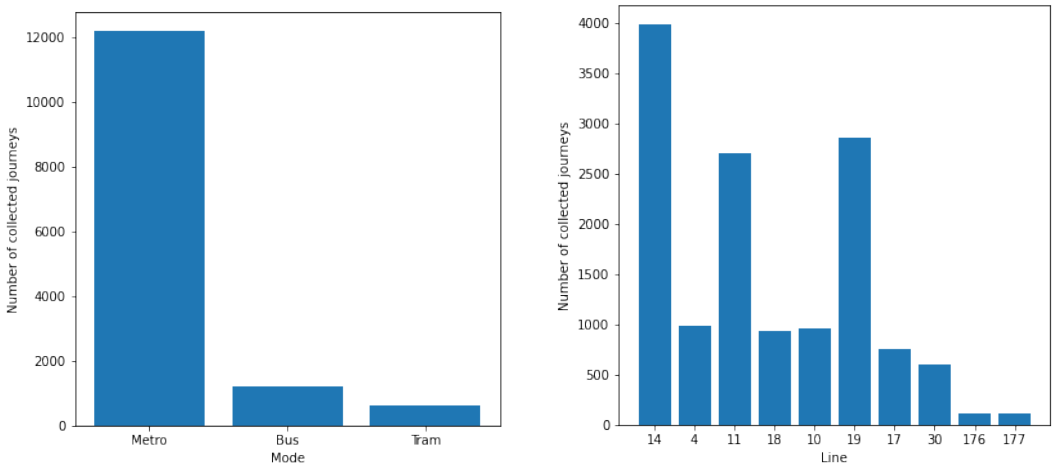


Figure 3: Distribution of collected data over time in the afternoon

3.2 Data processing

3.2.1 Adding the real-time arrival to data

The first thing we did with the data was to compute the real delay of the vehicles. In the data, we have access to the label "TimeTabledDateTime" and to the "ExpectedDateTime" one for any incoming vehicles. By making the difference, we got the expected delay, but it is the expected one because we have no information about the future real arrival time. We needed to make the most precise estimation of this one. Our approximation was to consider that the last "ExpectedDateTime" value for every vehicle in every station is a good approximation of the real arrival time. We could track a specific vehicle thanks to the label "JourneyNumber" which gives a vehicle a unique value for a given line and a given day.



(a) Distribution of collected data over the different modes of transportation

(b) Distribution of collected data over the different modes of transportation

Figure 4: Line and mode distribution of the training set

Since we performed a lot of API calls (one every 20 seconds when there is no error), we have in the worst case, the estimated time 20 seconds before the arrival time, low enough to consider that there is no major difference. Now, we had 3 different time labels for each journey: "TimeTabledDateTime", "ExpectedDateTime" and "RealDepartureTime".

3.2.2 Reconstruction of a journey

Now, we knew the real-time at a station, the goal is to link the stations together. At this stage, we have made no selection regarding the lines to study. We let the algorithm find all the links between the stations. To that extent, we go through all stations, collect the names of the different lines (and at the same time, we did the same with the label "JourneyNumber" to track the vehicles individually), and compare them with all other stations. By crossing the information, we were able to make the list of all vehicles that passed by 2 different stations of our study. This job was first made on one day to get all the interesting lines and then applied to the whole dataset, which was preprocessed removing non-relevant lines (and saving a lot of computation time). By concatenating the information from the departure and the arrival stations (the order can be deduced from the label "TimeTabledDateTime"), we can reconstruct a journey, as presented in Table 1.

Key		Line	Departure Station	Arrival Station	Departure Time-Tabled Time	Arrival Time-Tabled Time	Real Departure Time	Real Arrival Time
Day_value_1	JourneyNumber_value_1	...	...	...	...	...	...	...
Day_value_2	JourneyNumber_value_2	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Table 1: Journey representation DataFrame

3.2.3 Determining the delay before departure

With Part 3.2.2, we had access to the delay accumulated during the trip by a vehicle. However, we analyzed also the delay accumulated before the departure. For each vehicle at each station, we estimated the difference between the final delay (the difference between "TimeTabledDateTime" and "RealDepartureTime") and the estimated delay (the difference between "TimeTabledDateTime" and "ExpectedDateTime"), what can be called the "Future delay". We need time segmentation to be able to compare the different values. We took intervals of 20 seconds. This choice is relevant since it corresponds to the same time interval as the API. So it is the lowest value for which we can have a dense dataset. The reference time here is the difference between the "ExpectedDateTime" and the "RequestTime" (when we called the API). We made this choice because that corresponds to the expected waiting time for the traveler and it is the best information in possession of the traveler. To complete Table 2, we proceeded similarly to Part 3.2.1, by taking the last value of the "Future Delay" that respects the condition of a waiting time superior to the limit for each threshold.

Line	JourneyNumber	Station	Future delay when the expected Waiting Time is X
...	...	...	...

Table 2: DataFrame collecting the Future Delay for a given expected Waiting Time (for values of X between 0 and 1200 seconds with steps of 20 seconds)

3.3 Itinerary framework

3.3.1 Class Itinerary

In Python, we developed homemade classes to have a framework to have easier manipulation of the data. The first object that we created is the itinerary object. an itinerary is composed of a departure station, an arrival station, transfer stations (if there are any), and a series of lines. For instance, an itinerary can be:

- Departure station: Tekniska Högskolan
- Arrival station: Fridhemsplan
- Transfer station: T-Centralen
- Lines: [14,18]

For the same stations, different itineraries exist by modifying the lines. With the DataFrame with the journey that we constructed. We constructed for each of the 6 studied trips (Tekniska Högskolan from/to Fridhemsplan, Tekniska Högskolan, Globen from/to Solna Centrum) all possible itineraries. That pointed out some itineraries that we did not think about constructing the project (e.g. going from Tekniska Högskolan to Fridhemsplan with bus 4 and then

metro 11 to Solna Centrum), but we have the data to study them, so we included all of them in the possible itineraries.

3.3.2 Class Time-Itinerary

The second class we used was the timed version of the itinerary class. In this one, we added the time to the journey, meaning we added for each journey, the expected departure and arrival times and the transfer times. With a small front-end solution, we provided the user the following interface:

Itinerary from Tekniska högskolan to Fridhemsplan
Departure Time :16:04:30 Arrival Time :16:17:30
- In Tekniska högskolan, take line 14 at 16:04:30 to T-Centralen (arriving at 16:11:00)
- Walk 3 minutes and wait 0min30
- In T-Centralen, take line 11 at 16:14:30 to Fridhemsplan (arriving at 16:17:30)

That is the information that we want to complete for our different models.

3.4 Different itinerary-planner models used

For each of the models, we inspect the different itineraries (embedded with the Itinerary Class) and create (when it is possible) a timed itinerary with each model. Finally, we sort by the expected arrival time to finalize the list of the timed itineraries.

3.4.1 Model 1: Timetable model

In this first model, we only take into account the scheduled trips and consider that the vehicles are all of them on time. The different departure and arrival times correspond to the scheduled times.

3.4.2 Model 2: SL

In this model, we simply translated the results from the SL API to our framework.

3.4.3 Model 3-4-5: Realtime model/median model/quantile 90 model

This model is doing the same as the timetable model but instead of working with the timetabled values, we considered the real-time expected times. Moreover, we had some statistical values for the result. For a given generated timed itinerary, we added the information of the median arrival time on this itinerary, the quantile 90, by taking into consideration when the demand is made. We also added content about the probability of missing a transfer due to the potential delays of the previous journey and the potential delay of the next one.

This whole last operation requires our full preprocessing of the data. Values are obtained by concatenating the delays obtained in Part 3.2.2 and in Part 3.2.3. Since Table 2 is not full because inspecting 20 minutes earlier is not possible for the journeys that are at the beginning of the period of collection of data, we have added the distribution of the delays and not the individual delays. We sampled each distribution (to avoid too heavy computations) and did the addition of each pair of samples. This is an approximation of the sum of the distributions. To distinguish between the real-time model, median model, and the quantile 90 model, we sorted by the adequate value (Expected arrival time, median expected arrival time, and quantile 90 arrival time). The interest to do 2 different models is that the median model will penalize an itinerary that is often delayed, whereas the quantile model will penalize a model that has sometimes high delays.

3.5 Evaluation of the models

To evaluate and classify the models, we have made two experiments. Both are made on data that were not used to train the model. We assigned, for each model, the itinerary that is considered to be the best. Then, we computed for each model the real arrival time based on the collected data and compared these values. For this computation, the JourneyNumber is not taken into account here. We have supposed that the user takes the first vehicle of the specified line that arrives at the stop (it can be the one after if the first journey was delayed). We considered as reference times, the times when we called the SL API. We explored two situations:

- planning a trip that starts immediately
- planning a trip that starts in 10 minutes

Unfortunately, the SL API only provides information for immediate start, so it was not included in the second experiment.

4 Results

4.1 Real-time planner

In the first experiment, we got the results from Table 3, 4, and 5. Table 3 represents the proportion of the time a model outperforms another one. For instance, Model 3 gives a better itinerary (meaning that following Model 3 recommendation, the user arrives earlier) 11.52% of the time that Model 1, but Model 1 does better than Model 3 8.15% of the time. We do not reach 100% by summing because they may purpose the same itinerary. The differential proportion is given in Table 4. In the previous example, Model 3 is better than Model 1 11.52% of the time that Model 1 and worse 8.15% of the time, so the differential proportion is 3.37%. Table 5 presents the average difference in the arrival times. For instance, the itinerary given by Model 3 arrives on average 7.4 seconds before the one given by Model 1.

Model A / Model B	Model 1 (Timetable)	Model 2 (SL)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		7.62%	11.52%	12.45%	13.06%
Model 2	10.98%		16.34%	17.81%	17.85%
Model 3	8.15%	8.61%		4.05%	5.91%
Model 4	6.98%	8.13%	2.19%		2.31%
Model 5	10.43%	11.63%	6.84%	5.18%	

Table 3: Proportion of better proposition from Model A compared to Model B (real-time request)

Model A / Model B	Model 1 (Timetable)	Model 2 (SL)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		-3.36%	3.37%	5.47%	2.63%
Model 2	3.36%		7.73%	9.68%	6.22%
Model 3	-3.37%	-7.73%		1.86%	-0.93%
Model 4	-5.47%	-9.68%	-1.86%		-2.87%
Model 5	-2.63%	-6.22%	0.93%	2.87%	

Table 4: Differential proportion of Model A over Model B (real-time request)

Model A / Model B	Model 1 (Timetable)	Model 2 (SL)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		5.5	-7.4	-15.0	-4.1
Model 2	-5.5		-12.6	-20.7	-8.8
Model 3	7.4	12.6		-7.6	3.2
Model 4	15.0	20.7	7.6		10.8
Model 5	4.1	8.8	-3.2	-10.8	

Table 5: Average difference of Arrival Time between Model A and Model B (in seconds) (real-time request)

The different tables show that Model 4 performs better on average than the other models, regardless if we compare the number of better performances or the average arrival time. Surprisingly, SL model is the one that performs the worst. Using the median model allows the user to earn, on average, more than 20 seconds. This performance is even more outstanding when we note that 74.06% of the time, both models give the same itinerary. Thus, when the itineraries differ, the average advantage for the median model is more than 1 minute and 20 seconds!

4.2 10 minutes planner

In this experiment, we had to exclude the SL model since we cannot have predictions in advance. In this section, we have considered the itinerary that was advised 10 minutes before starting the trip. The results are given in Table 6, 7, and 8.

In this experiment, Model 4 is still the one with the best results in terms of both proportion of victory and average arrival times. However, the average saved times are lower than in the first experiment. This is a bit surprising, because we may have supposed that forecasting the delays more in advance would ameliorate the final prediction. However, in our

Model A / Model B	Model 1 (Timetable)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		9.15%	11.06%	11.98 %
Model 3	5.23%		2.43%	4.13%
Model 4	6.40%	1.69%		1.91%
Model 5	8.84%	4.86%	3.39%	

Table 6: Proportion of better proposition from Model A compared to Model B (10 minutes request)

Model A / Model B	Model 1 (Timetable)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		3.92%	4.66%	3.14%
Model 3	-3.92%		0.74%	-0.74%
Model 4	-4.66%	-0.74%		-1.48%
Model 5	-3.14%	0.74%	1.48%	

Table 7: Differential proportion of Model A over Model B (10 minutes request)

Model A / Model B	Model 1 (Timetable)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		-7.9	-10.9	-3.7
Model 3	7.9		-2.9	4.2
Model 4	10.9	2.9		7.1
Model 5	3.7	-4.2	-7.1	

Table 8: Average difference of Arrival Time between Model A and Model B (in seconds) (10 minutes request)

computation, we did not take into account the fact that the traveler can miss the transfer and we considered only the statistical results on the final journey of the trip (for computation simplicity). Thus, some transfers are too tight and that is not taken into account.

4.3 10 minutes planner including delays in the middle of the trip

To include the last observations in the model, we made the same experiment on only 300 samples (for a time question) but this time, we considered the arrival time for each journey to be the median or the quantile 90. In this framework, only feasible trips are purposed. The results are presented in Table 9, 10, and 11. Model 4 is still the best. It is difficult to compare the values with the previous experiments since we have not used the same testing values. For Model 4, his average difference in the arrival times with Model 1 and Model 3 is slightly better (since Model 1 and Model 3 provide the same results as in the second experiment, we can compare the different versions of Model 4). We can note that Model 5 provides too reasonable itineraries, resulting in performing poorer than the others.

Overall, the study findings indicate that Model 4 (Median model) consistently outperformed the other models in terms of providing the best itinerary.

Model A / Model B	Model 1 (Timetable)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		9.15%	11.06%	11.98 %
Model 3	5.22%		3.36%	5.30%
Model 4	4.48%	2.99%		2.27%
Model 5	12.88%	14.77%	12.50%	

Table 9: Proportion of better proposition from Model A compared to Model B (10 minutes request)

Model A / Model B	Model 1 (Timetable)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		1.87%	2.24%	-6.44%
Model 3	-1.87%		0.37%	-9.47%
Model 4	-2.24%	-0.37%		-10.23%
Model 5	6.44%	9.47%	10.23%	

Table 10: Differential proportion of Model A over Model B (10 minutes request)

Model A / Model B	Model 1 (Timetable)	Model 3 (Real-time)	Model 4 (Median)	Model 5 (Quantile 90)
Model 1		-6.62	-12.34	16.84
Model 3	6.62		-5.72	23.56
Model 4	12.34	5.72		29.37
Model 5	-16.84	-23.56	-29.37	

Table 11: Average difference of Arrival Time between Model A and Model B (in seconds) (10 minutes request)

5 Conclusion

To tackle the challenge of providing the best itinerary planner, we created different models to generate itineraries. Some models are based on the schedule (real-time or planning) and some are constructed with the delays observed on the lines in the history. For the latter, we needed to collect and analyze a large amount of data to be able to give some relevant statistical values. We implemented some new classes for Python in order to manipulate data easily and tested the different models on the data collected during one day. To evaluate the models, we compared the frequency that a model outperforms another, but also the average time saved by following the itinerary of a given model.

The results are unanimous. Whether for immediate departure or departure in 10 minutes, the model, that we constructed, based on the median delay outperforms the others. We expected even better performance for planning trips 10 minutes in advance rather than for real-time planning, but it was not the case in the experiments we conducted. The real-time planner is also quite robust. It has the advantage of not requiring much data to be constructed. On the opposite, the predictions given by SL are the most disappointing ones.

We can conclude that it is possible to create a planner that performs better than the SL solution. Whereas some studies focus on the solution to integrate the delay in the model

for the operator[5], we provided a solution for the users to adapt their itineraries with the knowledge of the potential disruptions of the network.

6 Remarks/limits

We have spotlighted some limits in our study:

- The accuracy of the SL API was not tested and some collected data must be not exact. This was visible especially when we sometimes got from the API vehicles that should have been gone some seconds ago (regarding to the "ExpectedDateTime"). In this situation, we did not change the way we attribute the "RealDepartureTime". Similarly, we have not distinguished the arrival and departure times for a vehicle in a specific station. The API does not provide enough information on that and most of the scheduled times are the same for arrival and departure.

- Some lines are equivalents on certain portions. For instance, between T-Centralen and Fridhemsplan, there is no difference between Line 10 and Line 11. We did not take that into account and processed them as two different lines. It would have been interesting to confuse them to generate the itineraries and also for the testing part.

- As mentioned in the results section, we considered only the median and quantile values on the last journey and not on the full trip for the first two experiments. The last experiment deserves to be investigated deeper. We have, for instance, not included the delays of the departures. This is not too hard to include in the model, but it requires more time to build it. The last model, we have created, is safer to the extent that we forced the traveler to arrive before the planned departure of the second journey.

7 Further considerations

This work can be extended in different ways. First, the comments made in the previous part are good starting points to have more accurate results and/or better models. Also, the last experiment could be conducted on the full dataset for a better comparison.

The scope of the study can also be extended by looking at other itineraries, other periods of the day, other thresholds, etc... It would be interesting to do some specific analysis and compare the benefits of our solutions case by case and study the factors that make them efficient.

We have not compared our models with SL solution for the second experiment. However, SL's app or SL's website allows to prepare a trip in advance. It would be great to incorporate these values into the study.

References

[1] London's Crossrail: A Public Transport Gem?, December 2015.

[2] TrafficLab SL API 2. Sl stops and lines v2.0. https://www.trafiklab.se/api/trafiklab-apis/sl/stops-and-lines-2/.

[3] TrafficLab SL API 3. Sl route-planner v3.1. https://www.trafiklab.se/api/trafiklab-apis/sl/route-planner-31/.

[4] TrafficLab SL API 4. Sl real-time information v4. https://www.trafiklab.se/api/trafiklab-apis/sl/departures-4/.

[5] Maha Gmira, Michel Gendreau, Andrea Lodi, and Jean-Yves Potvin. Managing in real-time a vehicle routing plan with time-dependent travel times on a road network. *Transportation Research Part C: Emerging Technologies*, 132:103379, 11 2021.

[6] Erik Jenelius. Ah2173 lecture 3 - public transport demand.

[7] SL. Sl planner. https://sl.se/.

[8] Region Stockholm. Public transport - region stockholm. https://www.regionstockholm.se/om-regionstockholm/Information-in-English1/Public-transport/.

[9] Wikipedia. Transport in stockholm. https://en.wikipedia.org/wiki/Transport_in_Stockholm.