DBT 2022 - Project 14

Minimize traffic congestion

Davide Piombini
davide.piombini@studenti.unitn.it
Carlo Veneziano
carlo.veneziano@studenti.unitn.it

July 1, 2022

1 Introduction

The aim of our project was to design a big data system capable of receiving traffic data and managing the traffic lights of Trento in order to optimize the traffic flow. In the very first part of our work, we explored different potential data sources such as private companies and public institutions. In the end, we exploited the open source software Sumo for simulating the traffic in the area of Trento and collecting data with the related API Traci. It would have been interesting to work with real data, but they were not available for the city of Trento.

About the use of big data technologies, we used Kafka to decouple the output of Sumo with the 'analytic engine' of the application, deployed with Apache Flink for stream processing. In the end, Docker images of Kafka and Apache Flink are used to package the project

2 Literature and related works

Searching for case studies in real-time traffic light control we soon realized that the literature still lacks practical implementations (here one of the best). In addition, according to Amini et al. (2017) [2], literature on applying Big Data approaches in Intelligent Transportation System (ITS) is rather scarce and none of the approaches we surveyed focus on big data stream processing. This acknowledgment is quite surprising given the high potential of this combination. In particular, it has been quite demanding to find papers and works regarding the application of big data technologies in the control of infrastructures of the network, for example the traffic lights. Furthermore, many papers found online aim to predict the traffic in some crucial junctions on real-world road networks [1] rather than controlling in real-time the traffic lights to minimize congestion.

Our task was to implement a streaming processing of traffic data and intervene directly on traffic lights in a real road network, so, as suggested by Amini et al. [2] and Krajzewicz et al. (2005) [3], exploiting Sumo simulations seemed to be the most feasible option.

2.1 Case studies in Sumo

Tutorials available online about traffic optimization with Sumo present a common way of thinking (e.g. this project). The network loaded in the Sumo configuration file is extremely easy and does not fit at all with a real city. In addition, in many tutorials, some induction loop sensors were added just before a traffic light in the network file (e.g. this tutorial). This process can only be done 'by-hand' in Sumo, so it would be really time expensive for an entire city network.

Also, the implementation of a machine learning algorithm for detecting the best combination of traffic lights involves hard reinforcement learning techniques. Reinforcement learning requires high-level machine learning skills ad hardware power we do not have.

Hence, we came out with an innovative solution to decide which traffic lights to intervene on.

3 Architecture

In this section, we are going to expose the conceptual, logical and physical data models we thought about in early stages to define the project.

3.1 Conceptual, logic and physical data models

- **Conceptual.** The entities are traffic lights and vehicles in the road network of Trento. Each traffic light has an id, position and state (red, yellow, green). Each vehicle has an id, position, route, speed and the id of the incoming traffic light's id. The aim of the system is to control traffic lights in order to let the vehicles complete their routes in the quickest time possible.
- **Logical.** According to the conceptual model the crucial information is the number of stopped vehicles at traffic lights in the red state. With the proceeding of simulation steps, the situation changes in the network so the idea is to minimize the overall waiting time of vehicles in real-time operating traffic lights switching. The decision criteria will be explained in Section **??**.
- **Physical.** Using the Traci API we extract atomic data from each step and we organize them on our own in python dictionaries. These data are directly published on Kafka's topic ad they are not stored since it would have been useless. Even if we would have implemented reinforcement learning, it does not require data for training.

3.2 Data pipeline

The pipeline is shown in Figure 1. A Sumo simulation is started to generate the traffic flow in the road network of Trento. Then, we use a Python script to convey messages to Kafka on the topic 'vehicle'. At this point, we use Apache Flink as a consumer of the published Kafka messages. Flink serves also as an analytic engine that states which traffic lights have to change state. The result of the algorithm is published back on Kafka in the 'output' topic. These messages are read by Python which acts on the ongoing Sumo simulation via the Traci API.

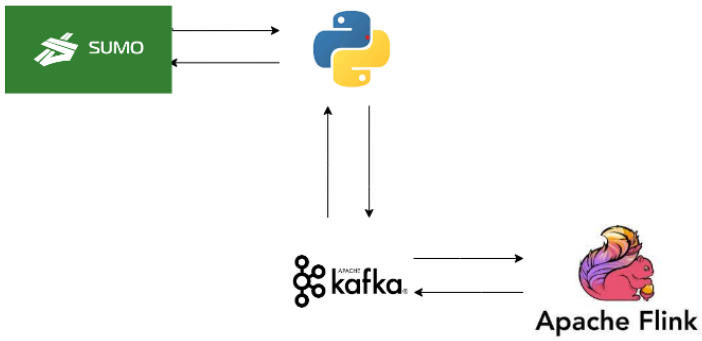

Figure 1: Proposed pipeline of the project.

The structure of Kafka's messages is exposed in the following pictures.


```
{
  "step": 22,
  "veh_id": "veh0",
  "speed": 9.042512049392192,
  "next_tl": "cluster_1935360399_2232255316_889555555_928552588_928552680_928552690",
  "next_tl_state": "r",
  "ts": 1656688090
}
```


Figure 2: Message on the 'vehicle' topic.

```
{
  "avg_speed": 0.0112422425,
  "tl_id": "cluster_1131515275_1311008983_260915132_430536801_9149862090_9149862100_935833
    366"
}
```


Figure 3: Message on the 'output' topic.

4 Implementation

4.1 Data collection

In the first phase of the project we started looking for traffic related data from any possible source. The aim was to find traffic lights data such as videos of traffic flows, phase's expiring time, average number of cars in line at traffic lights hour per hour.

As a matter of fact, nothing of these was freely available. By the way, the company Otonomo seemed to suit our needs, having an enormous amount of data taken directly from their sensors installed on vehicles all around the world (including quite a lot of data points in the city of Trento). We had a meeting with one Data Consultant in Otonomo who appreciated the importance of our project but told us that the data was in charge. The charge was about 100.000 €, as a minimums deal, so we discarded this way. Then we also approached the Google Maps API to get data related to traffic lights, but we soon realized that the Billing Google Account necessary for using the API costs 200€ per month.

Then we tried with the public institutions of Trento. As an example, we contacted the Mobilità Sostenibile bureau of Trento and the councillor responsible for Mobility and Ecologic Transaction Ezio Facchin asking for information about the existence of this kind of data. They replied us that traffic lights' data were not even collected. The Fondazione Bruno Kessler replied negatively too.

Finally, we discovered the opportunity of simulating the traffic on a real-world map using Sumo (short for Simulation of Urban MObility). Using Open Street Map we could import in this Sumo environment the map of Trento and then run over it a simulation with randomly generated vehicles.

4.2 TraCi API and Data Ingestion

TraCI is the short term for "Traffic Control Interface". Giving access to a running road traffic simulation allows to retrieve values of simulated objects and to manipulate their behavior "on-line". In the project we used the running simulation in Sumo as a server where to extract the data from. Hence, the data generation follows in real time the virtual simulation in Sumo. Step by step we connect to the Sumo API and take useful data for the analytic engine in order to make calculations and decide which traffic lights have to change their state. TraCi API has played a pivotal role in our project, but we also denoted some limitations in the resources and data retrieval functions. As an example it has been quite hard to fetch the vehicles blocked by each traffic light. As a matter of fact, in order to have a precise overview of the state of each vehicle in the network (cars blocked and ones not blocked by traffic lights) we had to couple by hand information that in real life would have been easily obtained through sensors, cameras or GPS positions.

4.3 Apache Kafka

We decided to use Apache Kafka as a message queue, in order to decouple the execution of the traci.simulationStep() function, that, as the name says, make the simulation going on, with the analytic engine, to have an almost real-time car flow in the streets of Trento. We used pykafka python library to manage both the producers and the consumers. To help us with bug-fixing, we added to our docker-compose file a (Kafka UI) that we used only to see the message flow. Kafka is not only used as the technology through which vehicles data are published, but also as the queue for the traffic light state changing messages coming from the analytic engine

4.4 Traffic light control algorithm with Apache Flink

The algorithm we deploy is quite really simple but effective. We used Apache Flink as a stream processor for the analytic engine of the application. To go a little deeper, we used the StreamTableEnvironment pyflink environment to read data from Kafka as SQL tables. Every 2 seconds, the algorithm

queries that table to obtain the last 4 seconds entries, group them by traffic light id, compute the average speed of the vehicles and considers only those with an average speed of approaching vehicles ¡= 2m/s. This query results are then stored into another SQL tables and published in Kafka in the 'output' topic. This messages are simply the ID of the traffic light and the average speed because of pyflink constraint of the 'where' clause. From the python script side, at each step, a KafkaConsumer consume that messages (from the 'output' topic) and simply changes each given traffic light state (through the id)

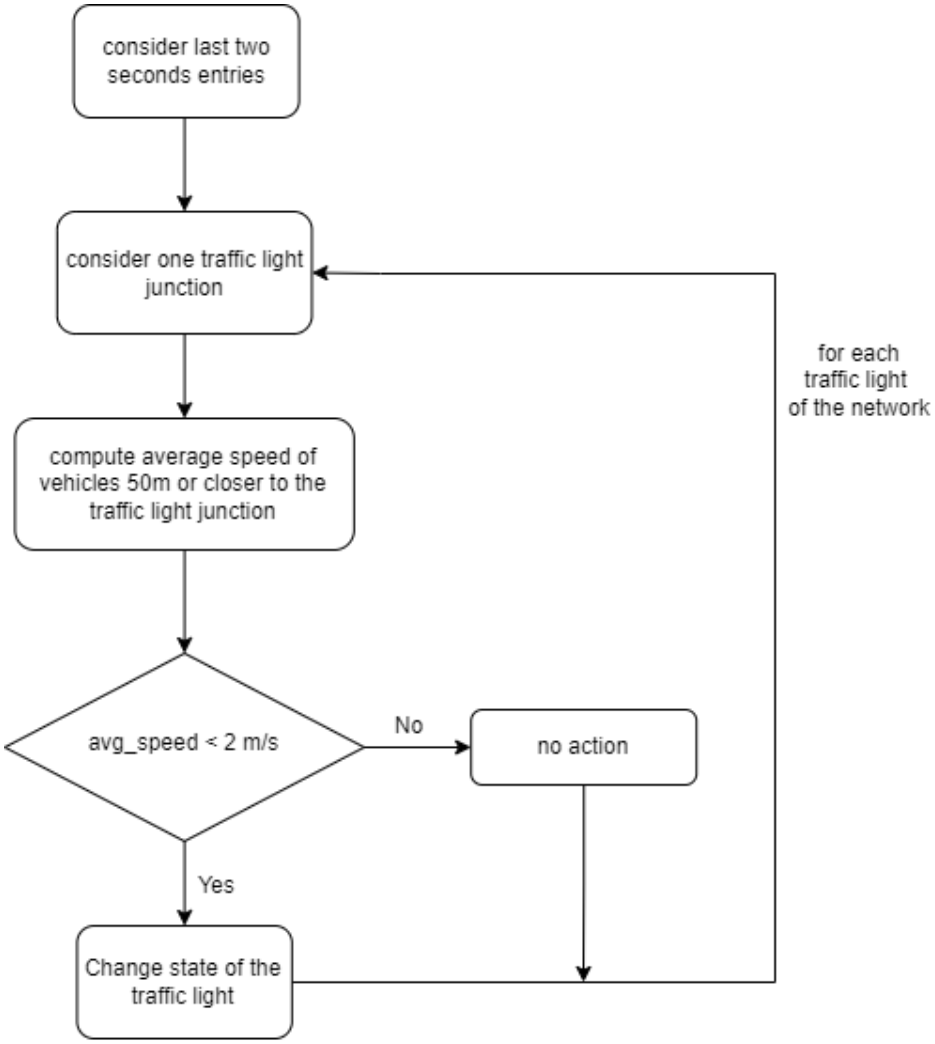

Figure 4: Decision algorithm flowchart.

5 Algorithm optimization results

In the project, we deploy a simple algorithm that runs for the whole simulation.

The way we measured the performance of our algorithm is by comparing the total network's waiting time when the algorithm is active and when it is not. In order to compute the waiting time we use the method of Traci 'traci.edge.getWaitingTime(edge)' for all the edges, namely roads, in the network. That method returns the time that a vehicle takes to cover the road, summed for all the vehicles on that road. To sum up we use the total waiting time as a proxy of the traffic flow's velocity.

Table 1 shows the total waiting time of the network in both cases the algorithm is applied and not applied, over a simulation of 1000 steps. Is possible to notice that the algorithm produces a decrease of 41882 seconds in waiting time.

Simulation	Waiting time (in seconds)
Standard	155400.0
With algorithm	113548.0

Table 1: Waiting time over the whole network.

6 Further improvements and conclusions

In the last stage of our work, we figured out some possible improvements in this kind of task. First of all, we acknowledge that Sumo is limited it simulates the traffic situation in a real street configuration. On the other hand, it does not reflect the actual traffic flow on the map (like data generated by real sensors on vehicles and traffic lights). The availability of real data could be undoubtedly useful to increase the importance of our project in the Intelligent Transport Systems field. In addition, in Sumo it is possible to integrate busses, bus stops, zebra crossings and so on.

Secondly, having a docker image of Sumo GUI would have made the project more user-friendly to display. Since this is not available, it is not possible to have a graphical interface that shows the traffic lights' state changes.

Thirdly, a reinforcement learning algorithm with accurate skills in the subject, as said beforehand, could increase performances by a lot.

Considering what has been said, we are happy with our results. The data-flow between important and innovative Big Data Technologies could be easily applied to reality. Even though the traffic light control algorithm we implemented is simple, we think that the result in terms of saved waiting time is a good result.

References

[1] Yazed Alsaawy, Ahmad Alkhodre, Adnan Abi Sen, Abdullah Alshanqiti, Wasim Ahmad Bhat, and Nour Mahmoud Bahbouh. A comprehensive and effective framework for traffic congestion problem based on the integration of iot and data analytics. *Applied Sciences*, 12(4), 2022.

[2] Sasan Amini, Ilias Gerostathopoulos, and Christian Prehofer. Big data analytics architecture for real-time traffic control. In *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 710–715, 2017.

[3] Daniel Krajzewicz, Elmar Brockfeld, Jürgen Mikat, Julia Ringel, Christian Feld, Wolfram Tuchscheerer, Peter Wagner, and Richard Woesler. Simulation of modern traffic lights control systems using the open source traffic simulation sumo. pages 299–302, 06 2005.