Electric Bus Simulator Project

Luiz do Valle

December 2020

1 Overview

This simulation platform was build in order to test the effect of different charging strategies on the bus schedule of New York City routes. The user specifies the length of time to run the simulation for, the technical specification of the bus and charger, how many chargers at each stop, and the graph representation of the route network and the model outputs three spreadsheets in csv format.

The first spreadsheet contains a row for each bus at each time step detailing the timestamp, the bus's ID, route, next stop, route direction, total distance traveled in Km, total energy used in KWh, the battery's State of Charge (SOC), and whether the bus is currently waiting at a stop. The second spreadsheet contains a row for each charging station in the network at each time step containing the timestamp, the name of the charging station, and a column for the ID each of the buses in the charging station's queue (the bus in the first column is the one currently charging while the rest is waiting to charge). Lastly, the third spreadsheet has a row for each timestamp listing the power output in KW of each of the charging stations in the network.

2 Data gathering

In order to make the simulation as realistic as possible, real-world data about the operation of New York City buses and the electric buses specifically was collected from a variety of sources from the internet and summarized in different places in the project. In this section, the original source for this data and the location in the project where it is stored will be listed.

- Bus route endpoints and length
  - Source (endpoints): http://web.mta.info/developers/developer-data-terms.html#data
  - Source (mileage): https://comptroller.nyc.gov/reports/bus-route-profiles/
  - Project location: data/stop_data/bus_route_endpoint_mileage.csv
  - Script to scrape: src/scrapping/Routes and Mileage Analysis.ipynb
- Stops by route
  - Source: https://bustime.mta.info/m/index?q=M1&l=&t=
  - Project location: data/stop_data/stops_by_route.csv
  - Script to scrape: src/scrapping/Routes and Mileage Analysis.ipynb
- Bus depots' address and total serviced buses
  - Source: https://www.ttmg.org/insidersguide/current-rosters/new-york-mta-bus-roster/new-york-mta-bus-roster-depot/
  - Project location: data/depot_data/depots.csv
  - Script to scrape: src/scrapping/Bus Depot Data Collection.ipynb
- Bus models at each depot
  - Source: https://www.ttmg.org/insidersguide/current-rosters/new-york-mta-bus-roster/new-york-mta-bus-roster-depot/
  - Project location: data/depot_data/buses_id_data.csv
  - Script to scrape: src/scrapping/Bus Depot Data Collection.ipynb
- New-Flyer Xcelsior CHARGE brochure
  - Source: https://www.newflyer.com/buses/xcelsior-charge/
  - Project location: electric_bus_manuals/Xcelsior-CHARGE.pdf
- Real-time electric bus location data
  - Source: collected using BusTime API
  - Project location: data/bus_time_log_data
  - Script to scrape: src/scrapping/bustime_scrapper.py
  - Analysis script: src/energy_analysis/Energy Analysis.ipynb

3 Stop network representation

To model the New York City bus stop network, a simple directed graph representation was used. The network consists of Stop objects connected by Edge objects. Each stop contains a mapping between route directions and Edge instances. Each Edge object stores another Stop instance and the distance between the current stop and the stored stop. When a bus is assigned to a route, it moves through the graph using the route direction annotation on the Edge objects of its current stop. The model also includes a Route object, which stores the Stop instances that belong to that route and the bus instances assigned to that route.

The simulation platform allows the user to load routes from csv files representing adjacency matrices. All the routes in New York City have been mapped this way and the adjacency matrices are available here. The Route Simulation jupyter notebook contains an example of how to load these representations into the simulation.

4 Bus driving efficiency model

To simulate the energy expenditure of a bus during operation, the simulation platform assumes that buses move at a constant speed and use a constant amount of energy per kilometer. These coefficients can be different for each bus in the simulation. Between each stop, buses move with this constant speed and consume this constant amount of energy for every kilometer traveled. Therefore, the speed and energy use per kilometer are two parameters that must be passed to the program before the simulation starts.

To find the average speed of the electric buses in New York City, the real-time location data mentioned in the Data gathering section was used. To obtain speeds from the raw latitudes and longitudes in the dataset, the coordinates of consecutive timestamps corresponding to the same bus were converted to distances. Using these distances and the time difference between the timestamps, it was trivial to calculate the "instantaneous" speed of the buses between timestamps. From this, the average bus speed on all observed routes over the period the data was collected was calculated. The result was an average speed of 10.33 Km/h with a standard deviation of 1.93 Km/h. So, 10.33 Km/h was used as the constant speed of the buses in the simulation.

The average energy use per kilometer was also obtained using the speeds calculated in the previous step. To convert the "instantaneous" speeds and accelerations of the buses to an energy use estimate, a set of electric bus dynamics equations was used. The equations can be found in section 2 of the following paper and are implemented in Python here in the BusModel class. The average energy consumption was found to be 0.627 KWh/Km with a standard deviation of 0.116 KWh/Km. So, the constant energy consumption of the buses in the model was set to 0.627 KWh/Km. This calculation only includes the energy consumed by the bus' motor while driving, excluding other energy expenditures.

5 Bus arrival and departure model

Each bus is assigned by the user to a route and starts at a stop and direction also specified by the user. Once the simulation starts, the bus moves between stops at a constant speed, as described in the Bus driving efficiency model section. When the distance between the bus and the next stop reaches zero (the bus arrived at the stop), the bus remains at the stop for an amount of time specified by the user (representing the bus waiting for passengers to embark/disembark). Currently, the bus does not spend energy while waiting at a stop but that can be implemented.

Once this waiting period ends, the bus has to decide to which of the stops connected to the current stop it must go. As described in the Stop network section, each stop has a group of edges that connects it to the stops adjacent to it. These edges are annotated with the direction and route that that edge belongs to. The simulation platform uses this annotation to determine where the bus should go next. The bus's next stop and distance to it are then updated and the bus starts moving once again.

6 Charging station model

Every Stop object has a list (that may be empty) of Charger instances, representing the charging stations. Every time a bus arrives at a stop, it needs to decide whether it can/needs to charge before proceeding to the next stop. The algorithm a bus uses to decide whether it will charge at a stop is described in detail in the following paper. A visual representation of the algorithm from the paper is also shown in Figure 1. A slight modification was made to this algorithm in that the bus is only released from the charging station once it charges to at least a certain SOC threshold (which can be set by the user). This was done to avoid the situation where the bus only charges enough to reach the next charging station and therefore stops at every charging station along the way.

Buses periodically stop at their bus depot for charging and maintenance. To model this situation, the simulation platform has special Stop objects that represent depots. These depots are connected to certain stops by edges just like any normal stop, but they are not part of the normal route stop schedule: buses only stop at these depots if they need to recharge there. When a bus arrives at a stop that has an edge to a depot that serves the bus' route, the bus uses the algorithm described in Figure 1 to decide whether to detour to the depot. If the bus does not need to charge or the depot does not have a charging station, the bus keeps going in its normal route. If the bus decides that it needs to go to the depot, it changes direction to move there in the following time steps. Once a bus finishes charging at the depot, it returns from the "normal" stop it left from and keeps going in the normal route. As of now, the existing electric buses in New York City can only charge at the Michael J. Quill Bus Depot in uptown Manhattan. This situation was modeled in the simulation platform by only

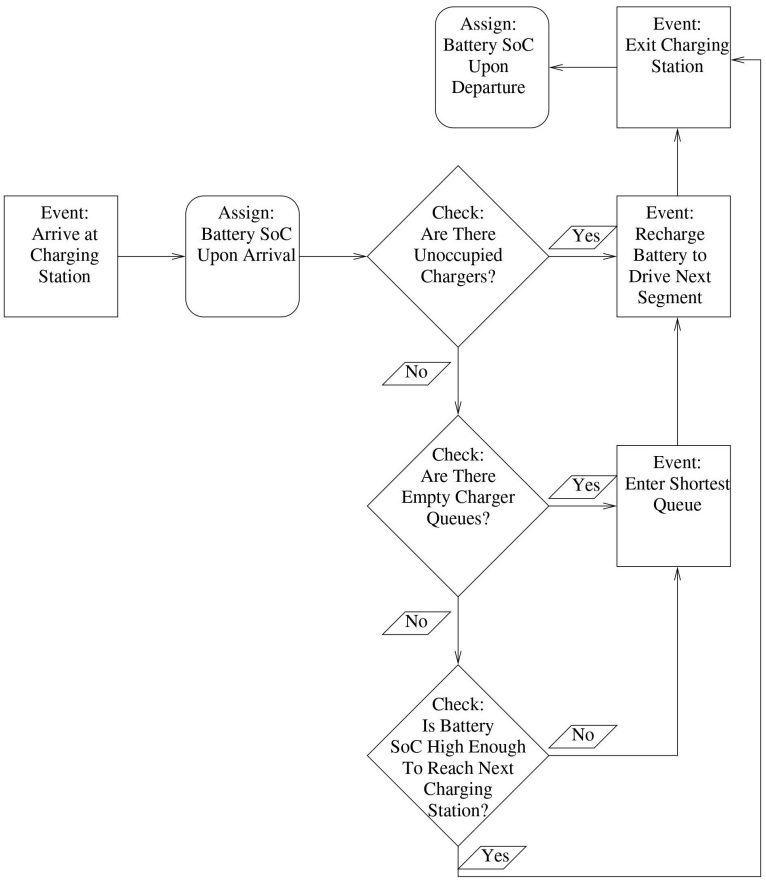

Figure 1: Charging algorithm for the simulation platform

adding chargers to this depot. However, since the effect on the bus schedule of the number of chargers at each stop and the charging algorithm is the situation being tested, these elements can be changed by the user.

In practice, the power output of a charging station decreases as the bus' battery charges because of the resistance to putting more charge in the battery. To model this situation, the Charger objects provide a constant output up to a certain SOC threshold (both the charging station output and SOC threshold are set by the user for each charging station). After this threshold, the charging station provides a linearly decreasing power output until it reaches 0 KW at 100% SOC. A plot of this piece-wise function is shown in Figure 2. In Figure 2, the red line represents the maximum power that the bus' battery can receive. If the charging station's power output is greater than this limit, the charging station will output at most the amount of power the battery can receive.

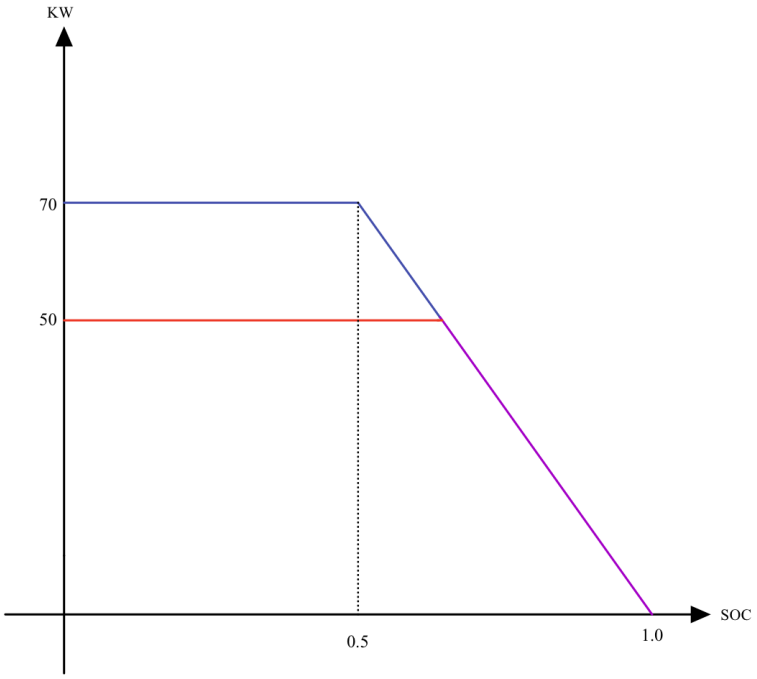

Figure 2: Charging function