

CS150 Project 1 Report

Oanh Doan

October 14, 2017

1 Introduction

The goal of this program is to project a growth curve for the number of charging stations as the number of vehicles changes. The underlying intuition is that more charging stations will need to be built to meet the growing demand for electricity, created by an increasing number of vehicles. To make the prediction, we first run a simulation to collect data. In our simulation, we make the following assumptions:

- There are no charging stations in owners homes
- when a car exhausts its battery, it is removed from the simulation
- each time unit of the simulation represents a minute of elapsed time
- each car has a unique id (auto-generated)
- each charging station has a unique id (auto-generated),

Furthermore, we simplify the problem by specifying that each station has only 1 charging point.

2 Approach

2.1 Design

There are 8 main classes in this program:

- Battery: contains properties of a battery and methods to check, decrease, or increase battery level as the car is traveling on road or charging at a station.
- RandomGaussian: generates random numbers from a normal distribution given a mean and a standard deviation.
- Car: stores properties of a car (id, speed, trip distance, trip interval, status, a list of charging stations etc.) and methods to simulate the behavior of the car. In the end, all of these methods are incorporated in *simulateOneStep()*, where the program will check various conditions to determine whether the car is at home, on road, or at a charging station, and performs the next step accordingly.

- CarList: is a car container, which allows us to simulate one time step for all cars in the list. It also contains certain print methods to perform the actions requested by the user through the console.
- Station: stores the station's id and its own CarList - a list of cars that are charging/waiting to be charged at the station. Once a car arrives at a station, the station manages the car by adding, charging, and removing it from the line.
- StationList: is a station container. It holds a list of stations and a random number generator to randomly generate the distances from a car to the all available stations when the car's battery is low.
- Program Input: reads user's input from a text file.
- ExperimentController: runs the simulation

In the program, we first create a list of stations. Then we create a list of cars, all matching the same station list. Thus, when any cars need to be refilled, they will look for a station from the same list. The only thing that is changing over time is the distance from a car to the stations, since this distance is randomly generated. As described in the Station class, once a car is in, the station has full control of the car.

In addition to the 8 aforementioned classes, we also create 6 unit test classes for Battery, Car, CarList, Station, StationList, and ProgramInput. There is no testing class for RandomGaussian because we cannot test methods generating random numbers. Figure 1 shows the result of our unit tests.

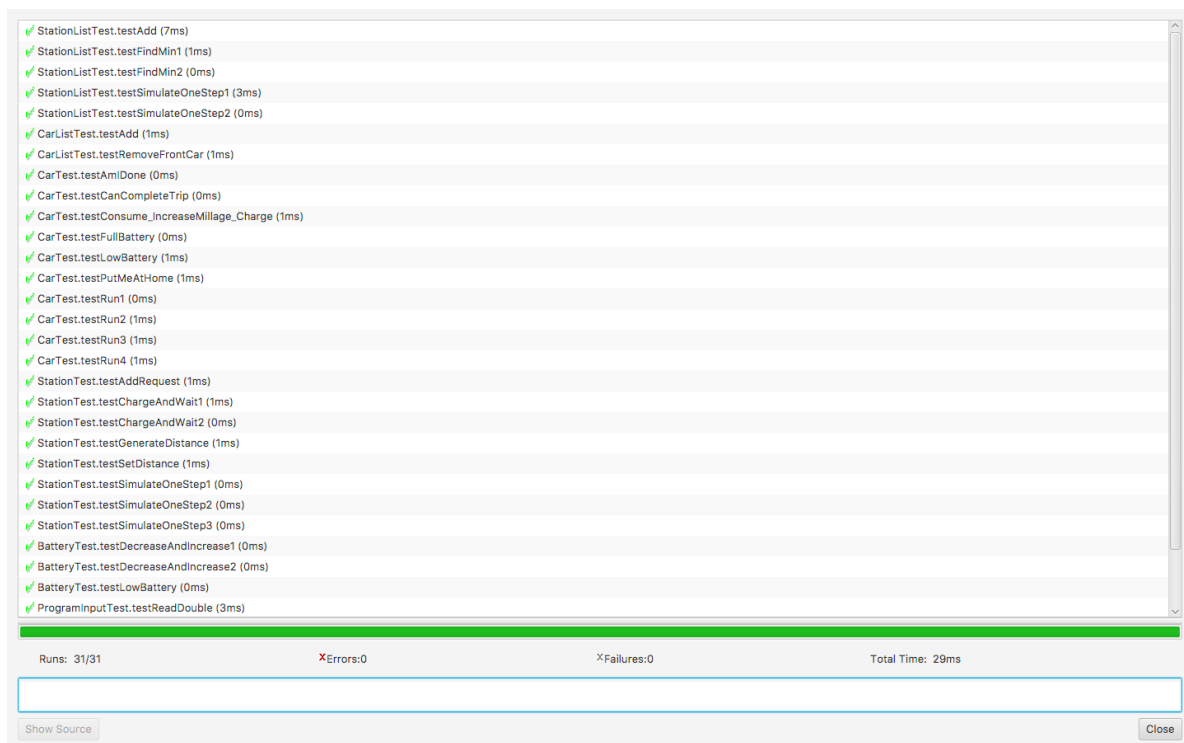


Figure 1. Results of unit tests

2.2 Choice of algorithm and Data Structure

- ArrayList (*ArrayList (Java Platform SE 8)*, n.d.): both of our car container and station container use ArrayList since it is easy to manage. Moreover, insertion at list end (add a new car/station) is fast ($O(1)$) and any other search/deletion algorithms takes $O(n)$ on average.

3 Method

- We fix the number of vehicles to be 100, and try 4 different numbers of stations (20, 30, 40, and 50).
- For each time step, we collect data on the total number of cars in the simulation, total number of cars at charging stations, and average waiting time. Note that we start measuring the wait time as a car enters the station, increment it every time step and record the value as the car leaves the station. The average waiting time in a time step is the average of waiting time across all stations in the simulation. Thus, there are steps when we see the average waiting time is 0. This means that either the station is empty, or some car is charging and has not finished yet.
- For each pair of number of cars - number of stations, we run 5 times with different random seeds to obtain "average" values for the total number of cars in the simulation, the number of cars at station, and the waiting time at each time step. In the end, we calculate the average of all waiting time values

4 Data and Analysis

Table 1 records the average waiting time of a system with 100 cars using different numbers of stations.

Number of stations	20	30	40	50
Mean of Waiting time	4.66	3.14	1.98	1.58
Median of Waiting time	4.66	3.14	1.98	3

Table 1: Average waiting time as the number of stations changes

From Table 1, we see that as there are more charging stations available, the average waiting time decreases. In addition, we plot the total number of cars in the simulation, the number of cars at charging stations and waiting time overtime. More specifically, we look into the cases when there are 20 and 50 stations in the simulation, respectively.

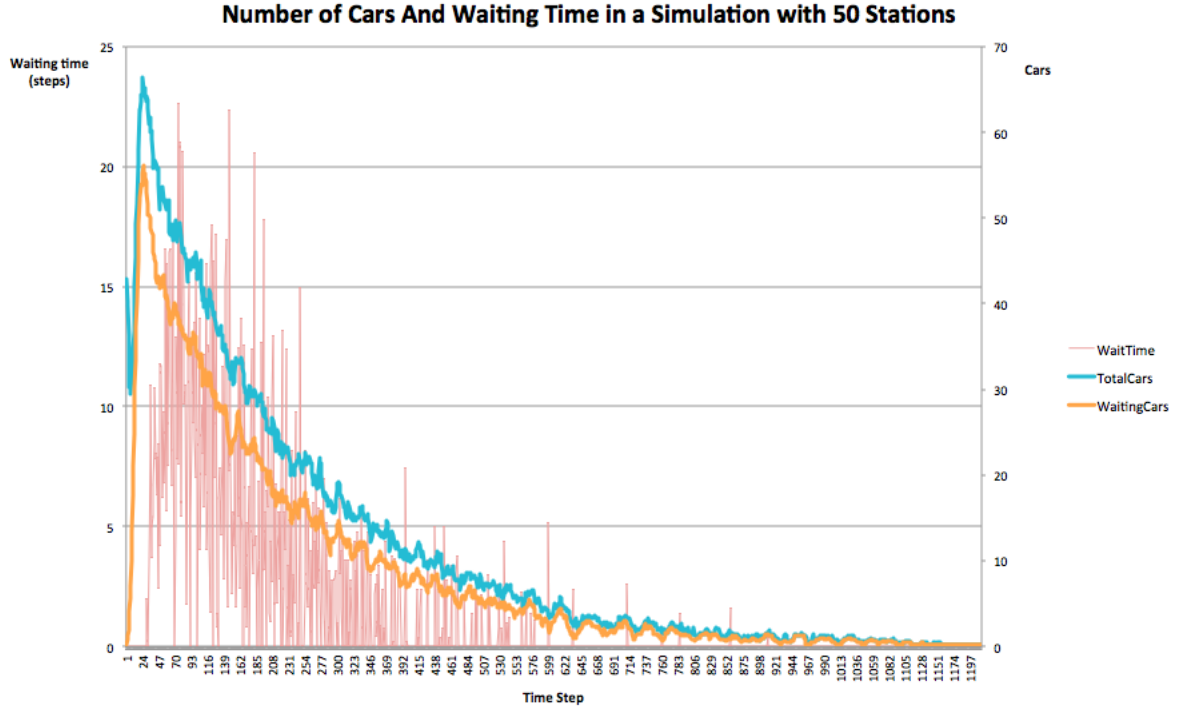


Figure 2. Simulation with 50 stations

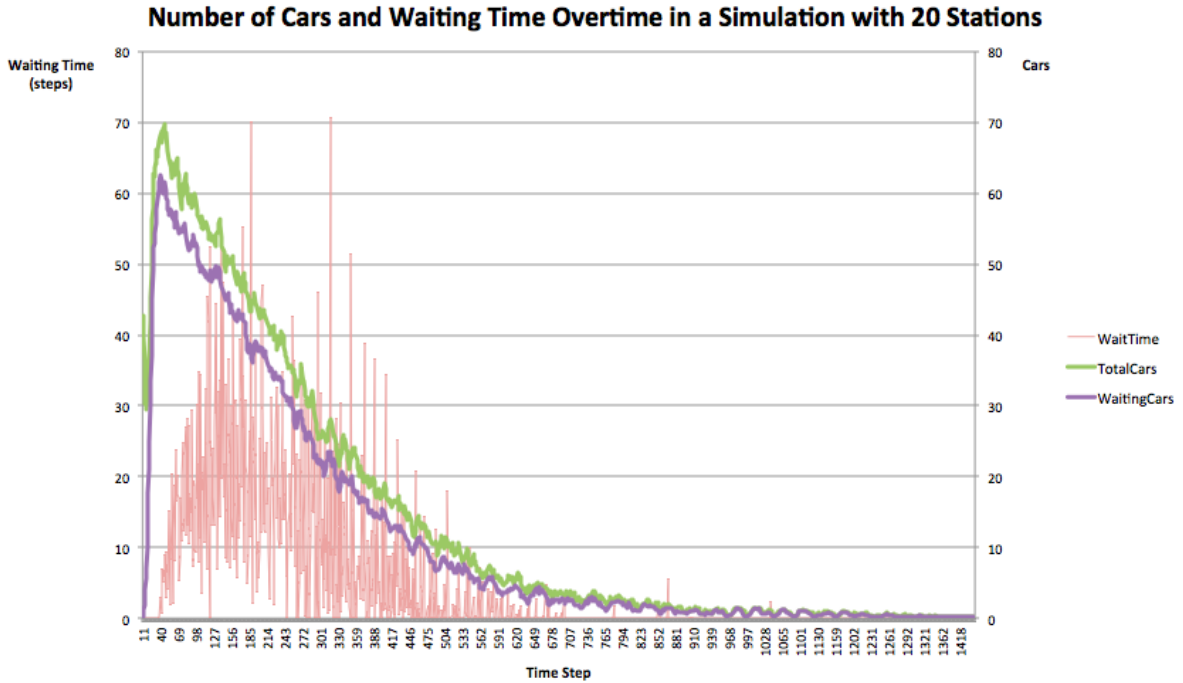


Figure 3. Simulation with 50 stations

Despite differences in scale, both Figure 2 and Figure 3 show that the number of cars at charging stations as the total number of cars increases. Furthermore, this growth leads to increased waiting time. As time goes by, cars die out, the total number of cars drop, and so

do the number of cars waiting at stations and the waiting time. However, we note that the peak of the number of cars does not align with the peak of waiting time. This makes sense because when a car starts a new trip, it takes a while before the car depletes its battery and needs to look for a charging station and line up there. In addition, we notice that even though both simulations have the same maximum number of cars, their maximum average waiting time differs. The maximum waiting time with 20 stations is three times as much as that with 50 stations. This observation agrees with our results in Table 1.

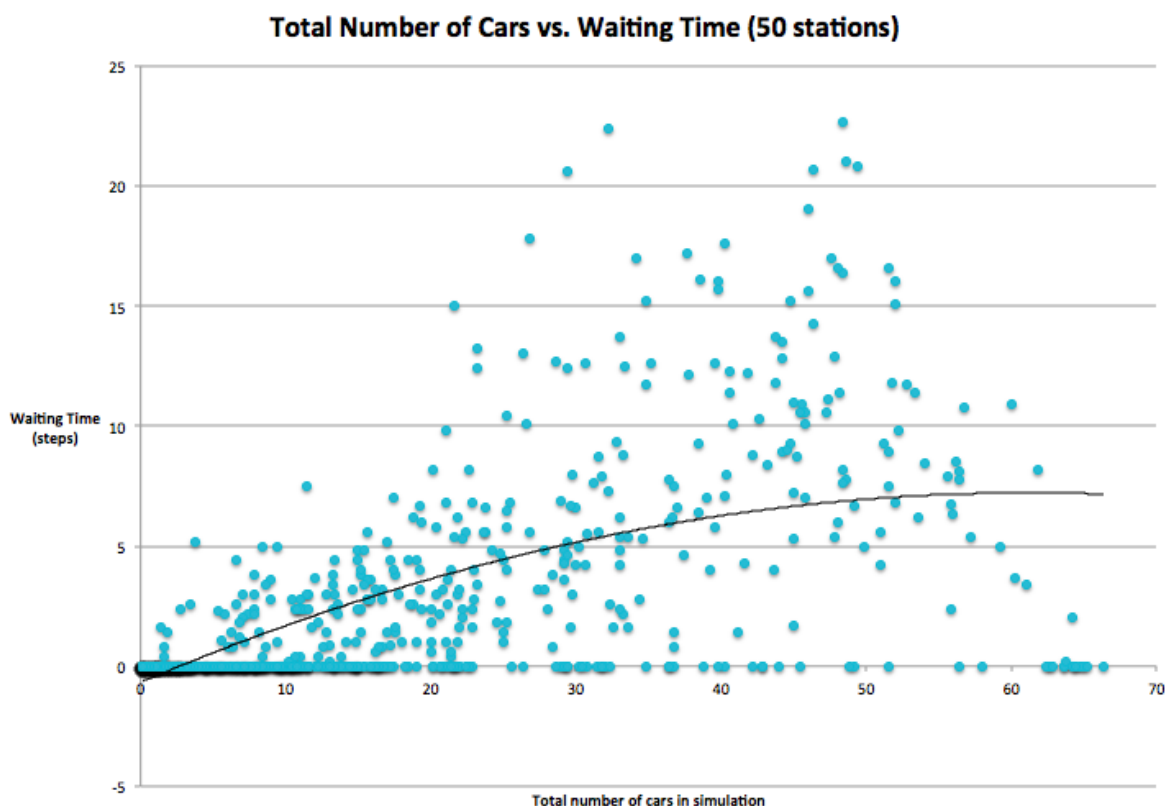


Figure 4. Total number of cars and waiting time (50 stations)

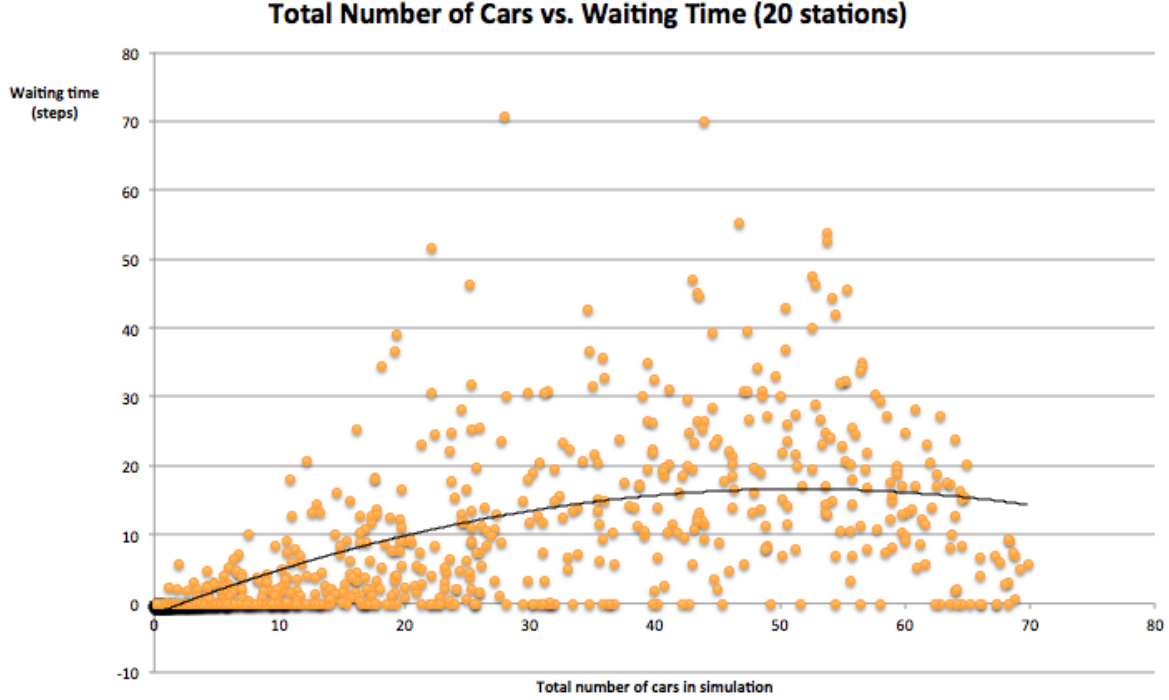


Figure 5. Total number of cars and waiting time (20 stations)

Figure 4 and Figure 5 plot the total number of cars and the average waiting time at each time step. Fitting a trend line through each of these scattered plots, we can clearly see that the waiting increases as more cars are in the simulation.

5 Conclusion

- Given a fixed number of charging stations, a growth in the total number of cars results in increased waiting time.
- With 100 cars, it is ideal to have about 30 charging to stations so that the average waiting time is about 3 unit time. We note that the distribution of waiting time is skewed right, so the average is actually lower than median. This means that during the most crowded time, the waiting time can be much higher as illustrated in Figure 2 and Figure 3, where the maximum waiting time is about 25 and 70, respectively.

References

ArrayList (java platform se 8). (n.d.). Oracle. Retrieved from <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>