Traffic Simulation Report

November 14, 2021

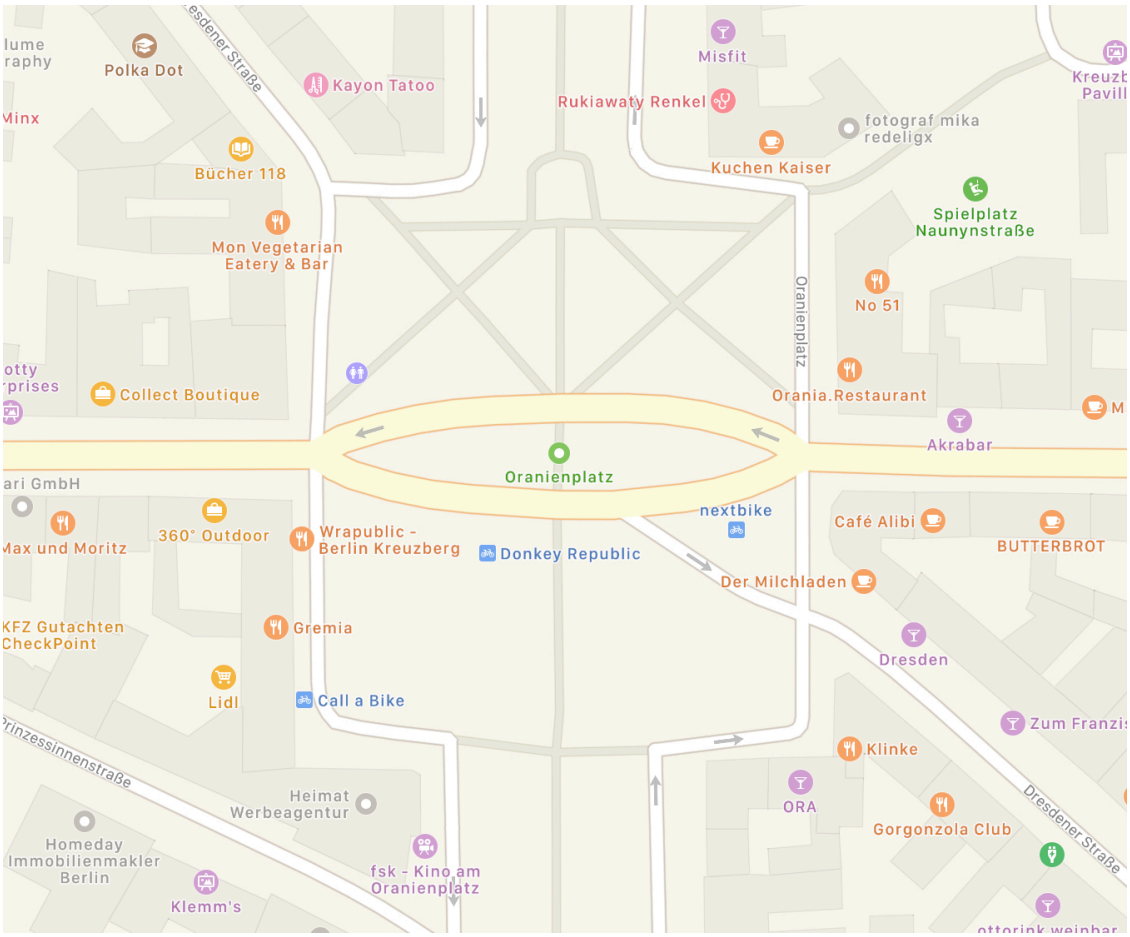
1 Traffic Routines to Improve Travel Times

1.0.1 Feedback and Grading

Bearing #audience in mind, the report below relegates technical implementation details to a short, high-level overview under 'Implementation'. In the zip file, I have attached two notebooks (`scratchpad.ipynb` and `graphsim.ipynb`) with stream-of-consciousness (read: messy) tests I performed while building the simulation, including testing the equivalence of spacetime diagrams of a 1D NaSch simulation implemented with numpy magic (as seen in class) and with the OOP approach I use below, flow tests for the graph implementation and others. I haven't figured out a `run pytest` workflow yet. Still, I spent most of my time performing sanity checks on the model, so tips there would be appreciated.

I also spent a reasonable amount of time designing what I (hope) is an extensible network-based model for traffic. Still, I fell short, in my mind, on interesting empirical data collection. Some ideas/suggestions there would be fantastic too!

1.1 Oranienplatz



For its history as the epicenter of the OPlatz pro-immigration protest and ongoing renown as a hotspot for fine cafés and cuisine, Oranienplatz has a relatively unsophisticated traffic system.

Compared to its sister-system Kotbusser Tor, of falafel and illicit SIM card-dealing renown, Oranienplatz has just two sets of traffic lights that coordinate traffic movements between Oranienstrasse, an arterial bilane linking the Kreuzberg neighborhood to the more extensive highway system, to Legeindamm Strasse, the twin single-lanes that link the nearest U-Bahns to the Turkish area.

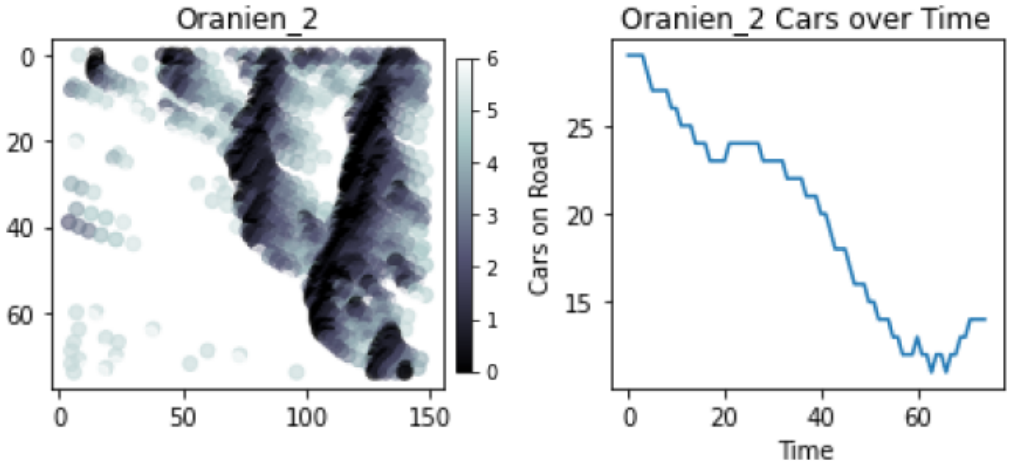
With a growing influx of residents around the neighborhood and increased congestion around late evenings, Oranienplatz needs an alternative strategy that coordinates its traffic lights. Such a strategy will improve the average traffic flow across the road system during its busiest hour, from 5 to 6 pm.

In this report, we implement a simulation of Oranienplatz using data recently collected from the square and explore how to use a graph-based simulation model to explore traffic strategies.

1.1.1 Simulation Overview

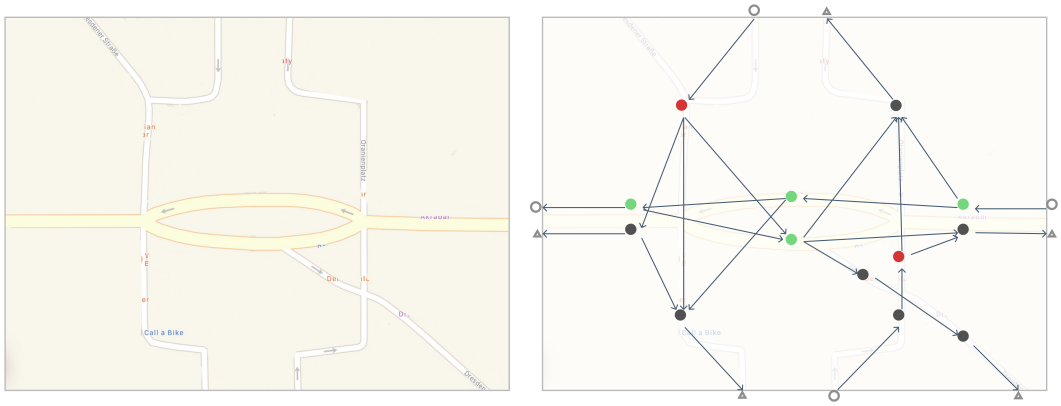
At its crux, the simulation implements a Nagel-Schreckenberg (NaSch) simulation. A cellular automaton, the NaSch model positions cars with integer velocities along a road of discrete cells,

moving cars along the road each 'step' (i.e. each second) according to a 'ruleset'. The ruleset captures the patterns of good driving behavior (drive below the speed limit, slow down if there is a car ahead) as well as the randomness in the real world (slowing down to not collide with a jaywalker). For a single road over time, the following spacetime diagram captures the traffic patterns that emerge:



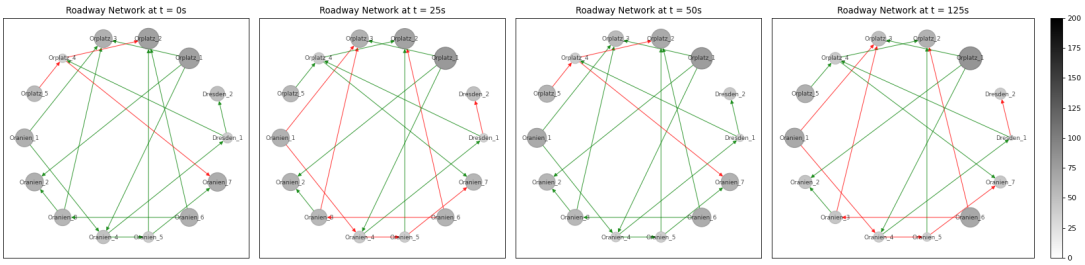
In the plot above, a segment of Oranienstrasse is simulated over 75 steps. Darker colors correspond to slower speeds, and dark patches indicate traffic jams drifting over the road, over time.

The simulation generalizes the NaSch model by allowing for multiple NaSch streets through an object-oriented description of each component (Cars, Lanes, as well as Generators for new Cars and Terminals for Cars exiting the simulation) and a network graph that describes the connections between these components. In short, Oranienplatz's streets are broken up into Road segments, and one segment is connected to another if there is a possibility to turn onto it:

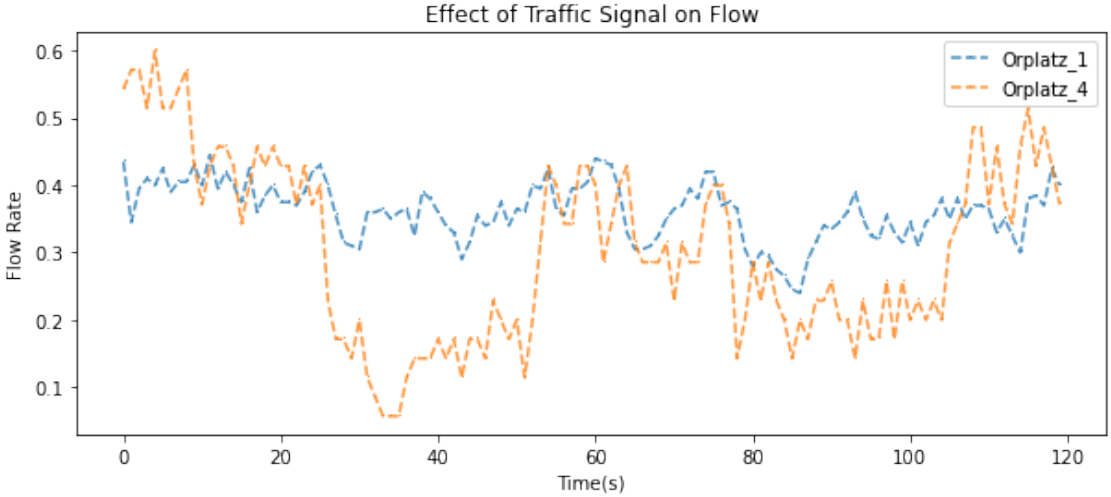


Every second, cars proposing to move past the end of the road segment 'turn' by moving from one node (the current road segment) to another node (the next road, chosen with fixed probability). Traffic Lights coordinate turning by labeling nodes 'Red' and 'Green.' If a Car is on a Green

node, it can choose outgoing nodes (ie, Roads) to turn to. The labeling works because two sets of intersecting Roads will have traffic lights that are *coupled*; there is a 'systolic' set of Roads and a 'diastolic' set, and where one set is Green, the other set must be Red:

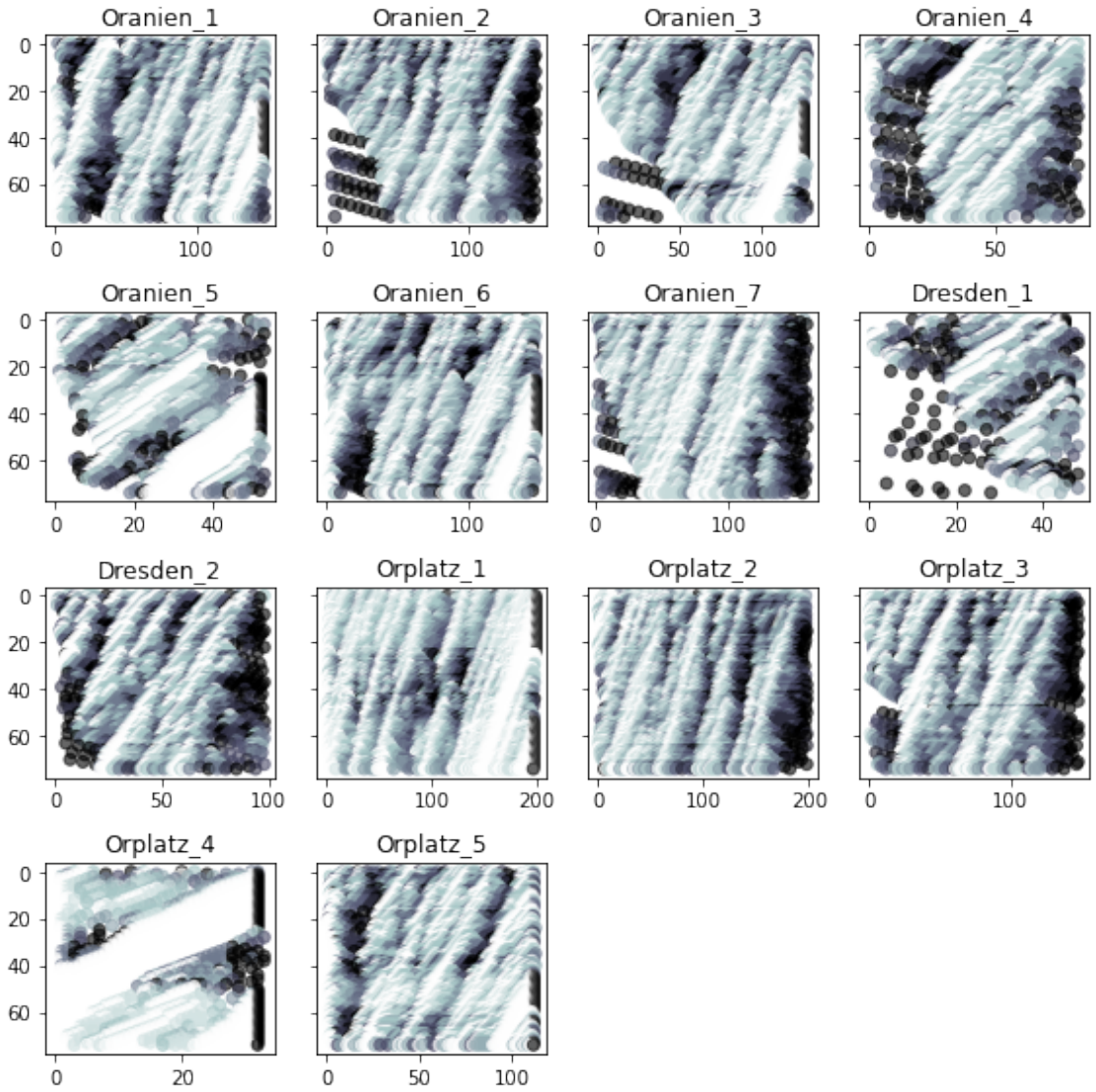


The figure above describes the alternating heartbeat of the two sets of traffic lights across 5 phases: red edges mean that cars cannot pass from outgoing nodes. In contrast, green edges imply that cars can. The phase of both sets of lights is 25 seconds, set to the empirical value described below. Additionally, node sizes scale by the number of cars in each road segment - notice 'Dresden_1' shrink over time as cars move to 'Dresden_2'.



Viewed another way, notice how the average flow rate - the key metric we will optimize for while considering other traffic light strategies - drops sharply in 'Orplatz_4' while the Road is marked Red. This low flow rate remains for about 25 seconds, corresponding to the phase of the traffic light. This is a broad indicator of a traffic pile-up on the road segment. Simultaneously, 'Orplatz_1' does not experience the same precipitous drops. This is indicative of a relatively empty road.

The simulation is bounded by 'Terminal' nodes (marked using triangles outside the map) which collect cars that exit the simulation. Additionally, cars enter the simulation through 'Generator' nodes (marked using bordered circles outside the map) according to an arrival rate inputted according to an empirical prior.



The plot above describes the simulation as a whole: cars entering the simulation, cars being handed from one lane to another, cars waiting at a stoplight, and cars exiting the simulation. The white band, about 25 seconds tall, within the 'Orplatz_4' corresponds to a Red phase that we described above. Straight black bars indicate cars waiting to transfer road segments, and the traffic 'waves' remain visible within each lane.

In summary, this is a grid-based, discrete simulation of how traffic patterns form over time. Its compactness of description allows us to test different traffic strategies through repeated trials quickly. However, its assumptions must also be laid bare:

1. 1-second Steps. Cars update positions and switch lanes every second. In practice, decisions on the road are not taken each second but are taken continuously.
2. Integer Velocity. Each car travels at an integer velocity and correspondingly accelerates and decelerates with integer values. In reality, we expect to see more smoothness in the transition of velocities: the simulation converts a smooth position-time graph for a car into a kind of step function.

3. Perfect Information and Good Behavior. Car behavior is modeled under the assumption of a kind of 'hyperawareness': each Car knows both the exact position of the next Car, but also their speed, its own speed, and the speed limit of the road.
4. Boundaries and Arrival Rates. As we cannot simulate the entirety of Berlin, the simulation is bounded using Terminal and Generator nodes. These Generator nodes have a fixed generation rate. Since the simulation is designed to simulate just one hour, this is a fair assumption but limits its extensibility to multiple hours, as we expect phases to change over the day.
5. Straight Lines. The simulation approximates roads as straight lines, with distances in the multiples of 5 meters. This is a fair approximation at the scale of a large city square, such as Oranienplatz, but not so much for smaller simulations with plenty of curved streets.

1.2 Data Collection

Between 5 and 6 pm, the following measurements were taken for each segment of the roads. Densities were approximated by considering how full road segments were over the course of the hour. Lengths, meanwhile, were approximated using Google Earth.

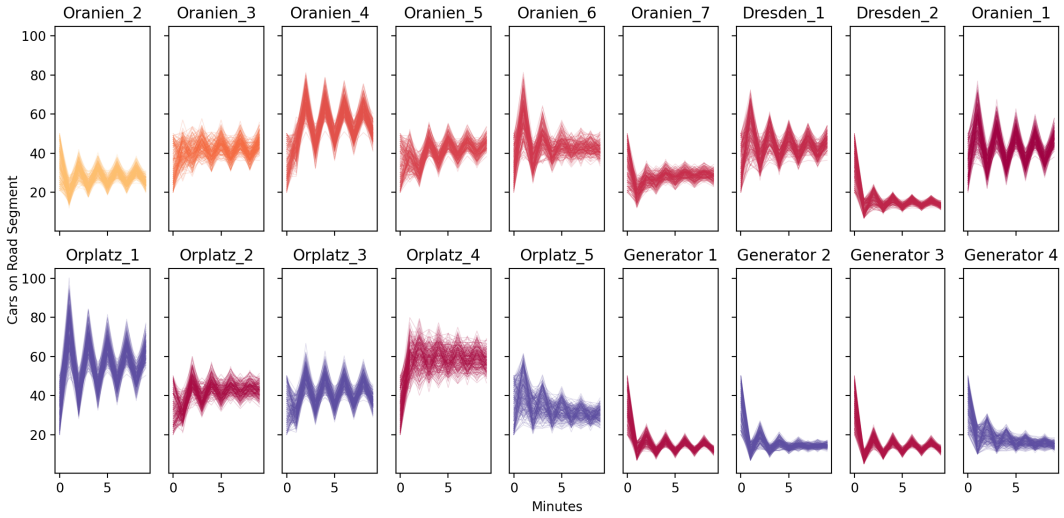
Road Name	Length(m)	Density (avg)	Speed Limit (kmph)
Oranien_1	150	0.4	20
Oranien_2	150	0.4	20
Oranien_3	130	0.4	20
Oranien_4	85	0.4	20
Oranien_5	55	0.4	20
Oranien_6	150	0.4	20
Oranien_7	160	0.4	20
Dresden_1	50	0.2	15
Dresden_2	100	0.2	15
Orplatz_1	200	0.3	30
Orplatz_2	200	0.3	30
Orplatz_3	150	0.3	30
Orplatz_4	35	0.3	30
Orplatz_5	115	0.3	30

During the same time, the number of cars passing by each generators were counted. Over a period of 5 minutes, cars passing by were recorded over 1 minute intervals, then averaged to obtain the arrival rate. It so happened that the arrival rate was constant at about 12 cars/minute, and this was set as a constant in the simulation.

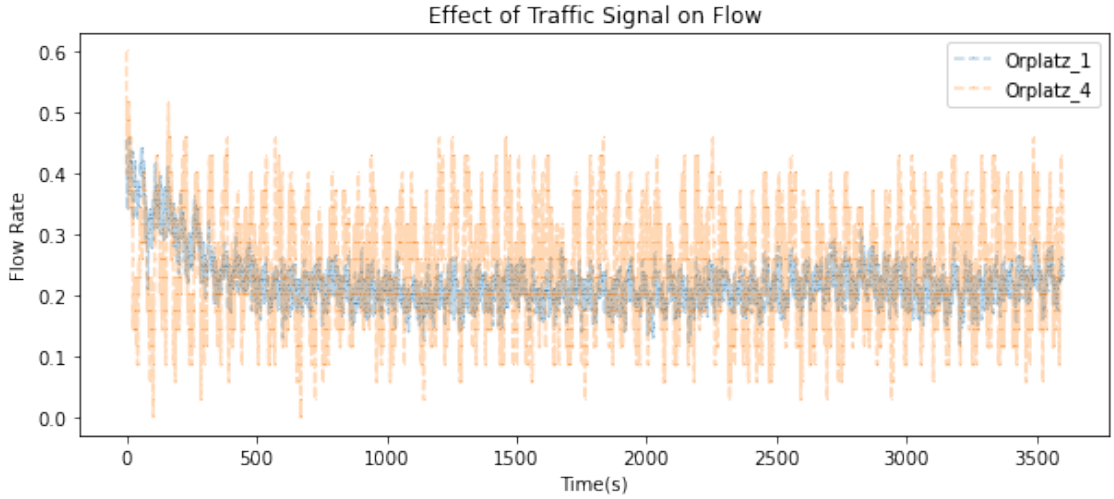
1.3 Theoretical Baseline

Before testing strategies, we need a theoretical baseline to benchmark results against. Earlier, we mentioned that a Car at the end of a green Road chooses randomly between the next nodes, owing to the fact that the model does not have priors or distributions over transit journeys. We can make use of this fact to represent the simulation as a Markov model, where an initial distribution u_0 of cars on each Roads is updated to a later distribution u_n by multiplying with the transition matrix $M^n u_0$. To set up this transition matrix, we first filter out Terminal Nodes (which simply collect cars), collect the graph's adjacency matrix, add the matrices transpose to itself, and then normalize

to form a valid Markov transition matrix. Crucially, we add the adjacency matrix to its transpose to create 'loops': these loops are what will give the distributions u_n periodicity, but also allow for a valid matrix that can be normalized. The stationary distribution, the eigenvector with eigenvalue 1, that the model converges to is summarized visually below:



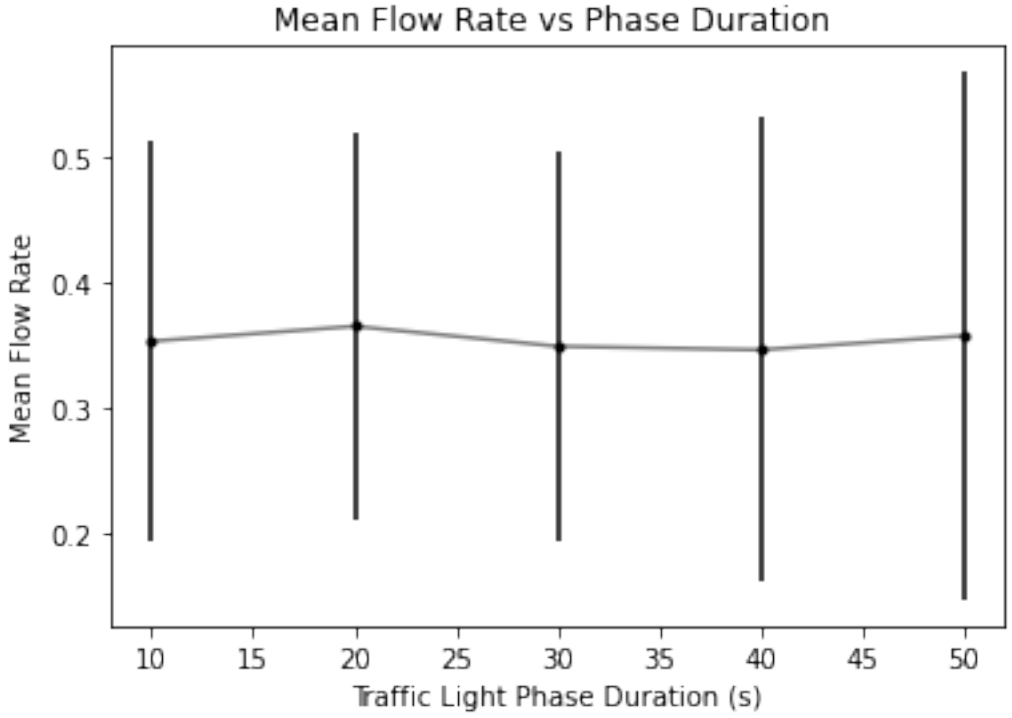
Initialized over multiple randomized starting distributions, the distributions portray interesting features. Generators, who simply generate cars and push then onto vacant Roads, converge to near emptiness, as we would expect. Oranien_4, as we have seen earlier, tends to fill up and stay full. Over a full run of 1 hour, this average flow rate plot shows promising correspondance between the Markov baseline and the simulation implementation:



These patterns, however, are more a consquence of the structure of the graph. Notice that the Markov model does not sufficiently capture the variation caused by the traffic phase.

1.4 Iterating Through Traffic Phases

We test out 5 different traffic light phases, ranging from 10 seconds to 50 seconds. To obtain better results, the simulation was repeated over 100 trials for each prospective traffic light phase, and the mean and confidence interval of the mean flow rate, across all lanes, is plotted. In each trial, we additionally sample only the last few minutes of the hour, to ensure that each lane has reached a 'steady state' behavior.



The plot evinces that our search strategy - iterating through different phase durations - has not yielded a clearly superior phase **when considering the confidence intervals**. For example, the original phase duration of 20 seconds has seemingly the highest flow rate, but the confidence intervals enclose all other phase durations. Notably, the confidence intervals seem to grow as we increase the phase duration, perhaps because pile-ups towards the ends of Red roads have a higher chance to form.

As such, we do not propose an alternative phase cycle with the results above. We suggest running the simulation with staggered phases, where the first set of lights is on one phase (25 seconds), while the other is on another, staggered phase (25 seconds, albeit 10 seconds after the previous set of lights).