PySimio: A Python Library for Discrete-Event Simulation

Kenta Takatsu (kt426)

Yuji Akimoto (ya242)

Omar Abdul-Rahim (oea5)

I. Executive Summary

In this paper, we aim to address the bus routing design need of the Tompkins Department of Going-Places. They run buses around Ithaca serving riders at three main stations, and they would like to improve their service. In particular, they would like to know how to split their buses in order to optimally serve arriving customers across all bus stops.

We approached this problem by building a simulation model. We built our model in Python because it gave us the flexibility to optimize based on more variable arrival situations. The Tompkins Department of Going-Places has some resource constraints, and there are general constraints of time, distance, and arrival fluctuation, so we thought our method to be the most robust given the circumstances.

We ran four primary experiments in order to test our simulation model. First, we ran the suggested initial test using the given fixed splits for buses. Second, we made our model slightly more sophisticated by only fixing bus splits on one route and allowing an optimization. Third, we further adjusted our models in order to allow for dynamic rerouting of buses to accommodate changing arrival rates. Fourth and finally, we used Bayesian optimization to get the best model results. This final experiment had the best runs out of all our simulations.

From our models, we found two models we recommend, broken down by three-hour intervals:

1. $[(4,1,2),(3,3,1),(4,3,0),(5,1,1),(3,2,2),(5,1,1)]$
2. $[(5,1,1),(3,3,1),(4,3,0),(2,4,1),(3,1,3),(5,1,1)]$

The first model was our best for minimizing average queue length and waiting time. The second model minimized extreme waiting times of two hours or greater.

II. Problem Description

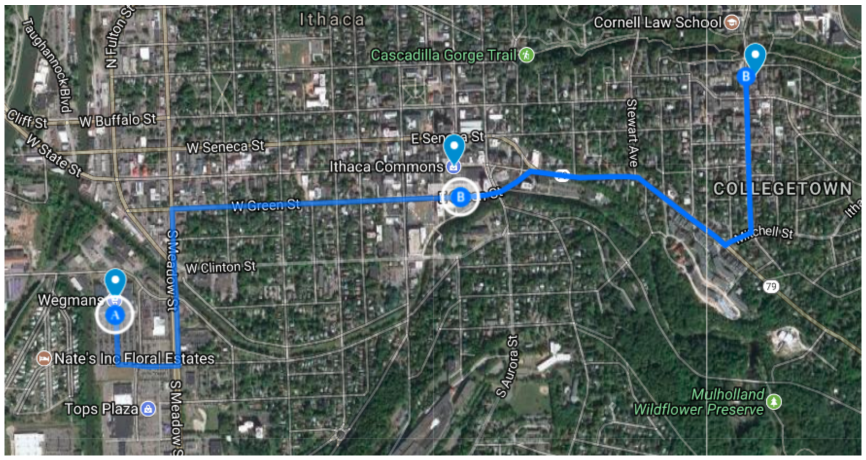


Figure 1

The Tompkins Department of Going-Places currently runs buses through Collegetown, the Commons, Wegmans, and their bus depot. They are looking to redesign their routing system in order to best serve the bus riders.

Buses run through six different stops distributed among the aforementioned destinations: one in Collegetown, one at the bus depot, and both eastbound and westbound stops for each of Commons and Wegmans. Riders arrive at the Collegetown, Commons, and Wegmans stops. A map of all possible routes that buses can take is shown in Figure 2.

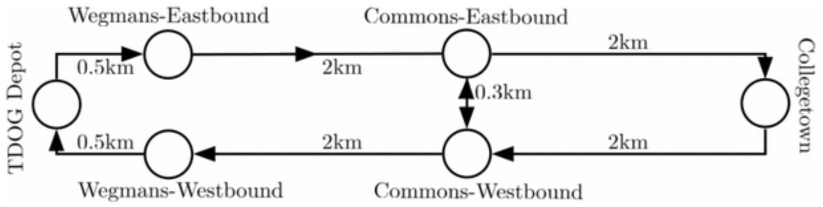


Figure 2

There are a few constraints on the scenario. First, Tompkins Department of Going-Places owns seven buses. Second, each bus has a seating capacity of 25 and a standing capacity of 10 (for a total capacity of 35 people). Third, there are several delay times that impact bus scheduling. In particular, each rider takes some amount of time to board, and each bus trip takes a variable amount of time depending upon the length of the trip leg. Fourth, buses run on three standard routes:

1. Starts at the depot, goes all the way to Collegetown, and then returns to the depot.
2. The Collegetown-Commons loop
3. Travels East from the depot till the Commons, and then returns to the depot

We have been tasked with creating a model that tells us which split of buses at which times best responds to the varying arrivals at the five stops and provides the best possible service to bus riders.

III. Modeling Approach & Assumptions

Because the goal is to optimize the service provided by the Tompkins Department of Going-Places, all aspects of the service must be considered in plan development and statistic collection. No single metric can quantify overall service quality, so we have come up with a list of important considerations for our model.

First, bus travel distance must be considered. There is an implied cost of running a bus service, and the source of cost most easily measured within our model is bus travel distance. Our assumption is that greater bus travel distance within a given span would mean higher costs for the Tompkins Department of Going-Places. Logically, the best case scenario would be one that minimizes the amount of distance traveled by the buses. We decided to track the travel distance per bus over all trips the bus took as well as total distance traveled in order to determine our model's worth.

Second, service utilization must be considered. It only makes sense to run a bus on a certain route at a certain time if it is actually being used significantly. For example, if there were a lapse in arrivals at the Wegmans Eastbound stop and demand to go to Wegmans declined for a certain time period, then it would not make sense to route buses in that direction. We are tracking the mean occupancy per bus as a metric for utilization.

Third, bus rider satisfaction must be considered. The service is not worth providing if people to not want to ride, and riders will not want to ride if what is available is not satisfactory, so this is worth exploring. We broke the problem of bus rider satisfaction into two parts: waiting time and in-ride comfort. For wait time, the amount of time a rider waits for a bus certainly impacts the level of satisfaction the rider has with the service. We are tracking the expected wait time for a trip, and the percentage of instances overall that a rider's wait time is greater than the expected wait time. For in-ride comfort, the enjoyment of the actual ride also impacts the level of satisfaction the rider has with the service, and in our opinion, ride enjoyment is most affected by whether or not a rider has to stand for their trip. We decided to track the mean number of people standing per trip, the percentage of trips that have an occupancy greater than the seating capacity of 25, and the percentage of trips that are at full capacity (the least comfortable situation). While this may seem to contradict our regard for service utilization, it is still an important component of our analysis.

Lastly, model feature efficacy must be considered. By this, we mean that it is valuable to know how much the route optimization features of our model are being used. Our model allows for mid-trip route changes (at all stops) as a response to demand around the map. We decided to track the percent of trips where routes have changed as a measure of how useful this feature actually is.

Initially, we believed that we should optimize waiting times and queue lengths. Our assumption was that a good bus service meant that riders would not be waiting in line for long amounts of time. However, upon further consideration, we realized that there is not necessarily correlation between queue length and waiting time or, by extension, service quality. Unless the queue is longer than an oncoming bus or group of closely-scheduled buses can handle, queue length has either no impact or negligible impact on waiting time. Thus, we decided that queue length was not a a sufficiently important factor to optimize. Rider waiting time is what best determines service quality overall, and every other aspect of service quality seems to stem from it in some way, so it is what we have chosen to optimize.

IV. Data Analysis

The arrival data shows patterns that are generally followed by bus riders. Examination of this data reveals the there are important routing considerations to be made. The estimated number of arrivals of riders by hour and by both initial point and destination point are shown in the heat map in Figure 3.

The most immediately noticeable characteristic of the heat map is the the large amount of arrivals from the Commons to Collegetown between hour 3 and hour 5 and the similarly large amount of arrivals from Collegetown to the Commons between hour 6 and hour 8. Upon further examination, there are also particularly low demands to go from anywhere to Wegmans during these time periods. It stands to reason that, in order to best meet demand and keep customer waiting times short, buses should be scheduled with more concentration on cycling through the Commons and Collegetown than going through Wegmans or the depot.

In addition, the arrivals on both ends of the day are relatively uniform with a slightly higher arrival rate in routes between Wegmans and Collegetown. This may be because people are trying to run errands or get to their service industry jobs either in Collegetown or at Wegmans. Once

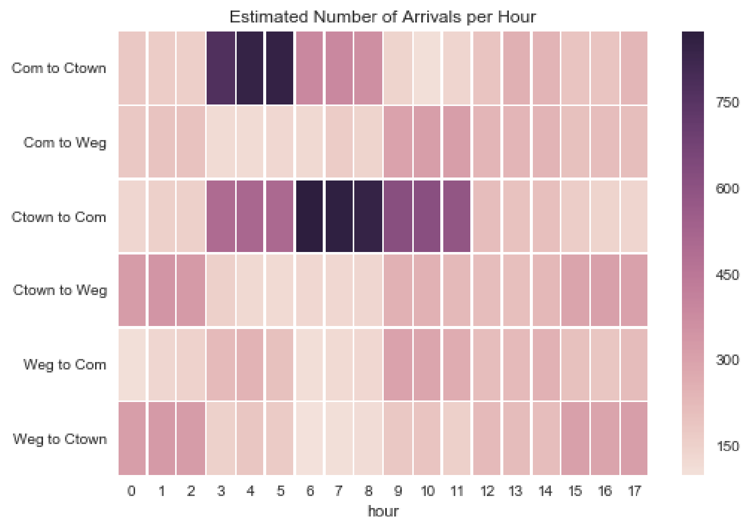


Figure 3

again, it would makes sense to favor routes with these stops slightly over others.

These initial assertions will not necessarily hold true for the model. After all, arrivals within the model follow a random process simply governed by known distributions. The only certainty we have is about value expectation, not actual value. Therefore, if the model does not follow what was previously stated, that does not necessarily mean that the model is incorrect.

Each bus route also has characteristics that need to be considered in model building. These characteristics are sometimes limiting factors in building the model, and they are sometimes they can be taken advantage of when planning routes. For example, the total number of arrivals of riders travelling from Collegetown to the Commons is shown in Figure 4. This route holds a notable property: the cycle on the route is short and only stops three times per cycle. Moreover, only two of the trip legs are prone to variance in travel time. The quick cycle time and relative stability of this route are useful properties to keep in mind. However, this route in addition to a route connecting only the Commons to Wegmans is not a good option for riders who want to travel from Collegetown to Wegmans. In other words, the route that makes the entire loop is necessary, even though it is a slow cycle that is prone to much more travel time variance.

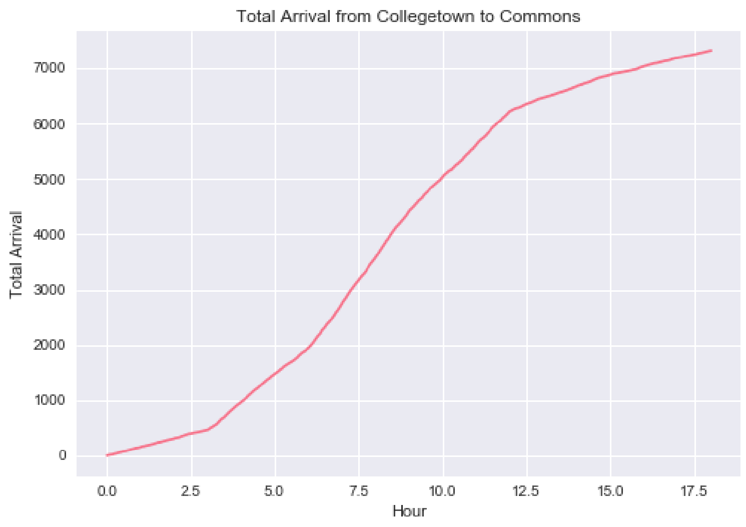


Figure 4

V. Model Verification

Our implementation allows for event-by-event simulation as well as overall statistic collection, which allowed us to verify the correctness of our model in the following ways:

Correct Boarding/De-Boarding All boarding/de-boarding procedures were handled in hard-coded if statements: a passenger would only board a bus if the bus' route includes the passenger's destination stop, and would only de-board the bus if the current stop matched the passenger's destination stop. All people were generated at their origin stop, and appended to the list of people waiting in order of their arrival time. The boarding process iterates through the list sequentially, and so each bus stop is a (constrained) FIFO queue. As an additional check, we verified that every bus is always empty upon arrival at the TDOG Depot.

Dynamic Re-Routing In our implementation, each bus maintains a list of stops that it travels between. Each route is a loop and therefore the next stop for a bus staying on the same route can be calculated by proceeding to the next item on the list, with wraparound. Our model also allows for dynamic re-routing, that is, buses can change their route at any time, and without having to return to the depot to do so. Since buses must be able to drop off all passengers before switching, we hard-code the stop that the bus must reach before changing its route - we verified this using a step-by-step evaluation, verifying that all buses travelled in the desired sequence for each of the 6 possible route changes.

Closed System To verify that people are generated consistently with the provided arrival rates and that people do not magically appear/disappear, we compute the total number of arrivals and departures recorded by the system. At the conclusion of a single day, the total number of arrivals should equal the total number of departures, and both of these quantities should be approximately equal to 28,575 for the sample arrival rates recorded by the TDOG.

VI. Model Analysis

In validating our model performance, we collect in total 70 statistics from each simulation. The breakdown of the statistics is as follows:

- Mean occupancy of each bus
- Mean people standing in each bus
- Mean traveling distance of each bus
- Time-series data of each bus occupancy
- Mean waiting time for each bus stop
- Mean queue length for each bus stop
- Time-series data of queue length for each bus stop
- Mean waiting time for each origin-destination pair
- Mean bus occupancy for every adjacent bus stops
- Time-series data of bus occupancy for every adjacent bus stops
- Total bus distance
- Total waiting time
- Total queue length

We mainly used 2 metrics, the mean queue length and the mean waiting time for all bus stops, to assess the performance of our models. For the parameter tuning, we investigated the time-series data of both queue length and bus occupancy. For the visualization, we used the Python open source, *seaborn*. Each plot in our time-series represents the 68-95% confidence interval of the given time of the day, and the box-plot uses every 25 percentile as a marker.

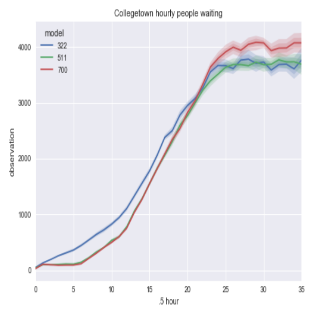
We conducted series of simulations to discover optimal bus scheduling for the TDOG. In the following section, we defined a bus schedule with X buses on Route 1, Y buses on Route 2, and Z buses on Route 3 as Model XYZ. For example, a model with 3 buses on Route 1, 2 buses on Route 2, and 2 buses on Route 3 would be referred to as Model 322.

I. Initial Analysis

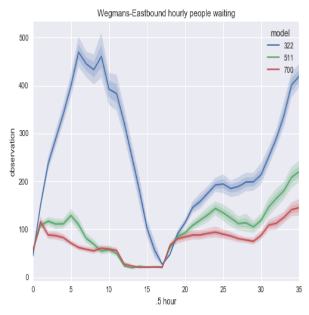
Our first simulation was the suggested initial trial. For this, the bus splits were fixed. The models given to us to use were 322, 511, and 700. Because of their fixed nature, running these models was easy to track. Moreover, certain types of problems could be easily attributed to particular splits. The graphs in Table 1 show this.

Table 1: Hourly Queue Length

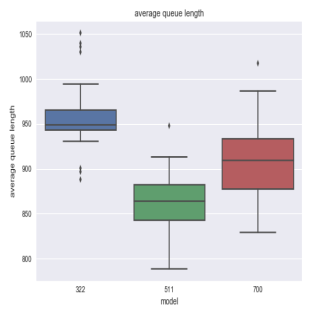
(a) Collegetown



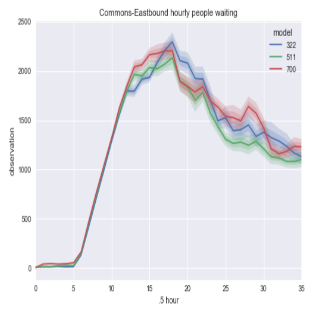
(b) Wegmans-Eastbound



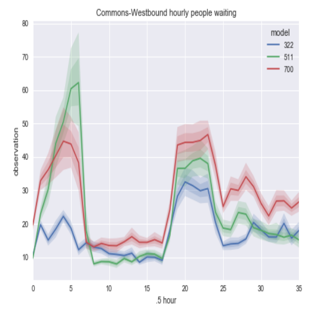
(c) Avg. Queue Length



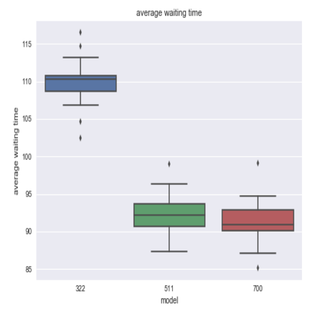
(d) Commons-Eastbound



(e) Commons-Westbound



(f) Avg. Waiting Time



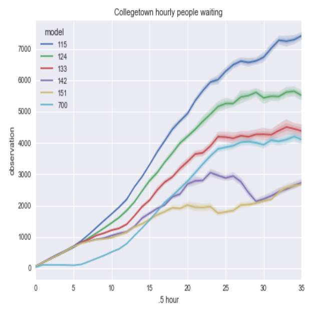
Graph (b) shows the number of people waiting at the Wegmans-Eastbound stop over the course of the day. The 511 and 700 models resemble each other somewhat closely, while the 322 model looks quite different. Because of the smaller number of buses operating on Route 1, the 322 model is unable to keep up with the demand of riders leaving Wegmans. This also prevents the 322 model from bringing as many riders to Wegmans as the 511 and 700 models, which explains why the lower number of people waiting in general at Commons-Westbound in graph (e). Graphs (c) and (f) track the mean queue length and mean waiting time for each model respectively. The 511 model did best for queue length, and the 700 model did the best for wait time. However, there is room for improvement across the board. We believe that because of Route 1's previously aforementioned inefficiencies, improvements could be made by altering the peripheral factors, leading us to our next attempt.

II. Route 1 Fixed

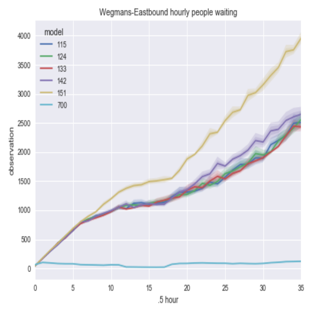
Our second simulation involved fixing the number of buses on Route 1 and trying other split combinations across the other two routes. The logic here is that there exists an optimal number of buses on Route 1, so it will be easier to find the optima based on the other factors when Route 1 is fixed. We first tested models for 115, 124, 133, 142, 151, and 700 as a baseline.

Table 2: Hourly Queue Length

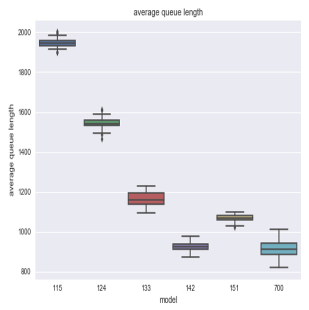
(a) Collegetown



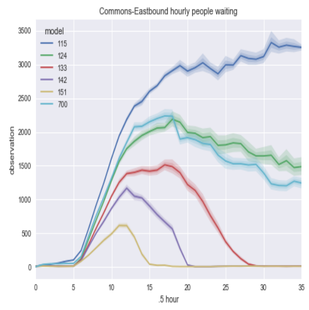
(b) Wegmans-Eastbound



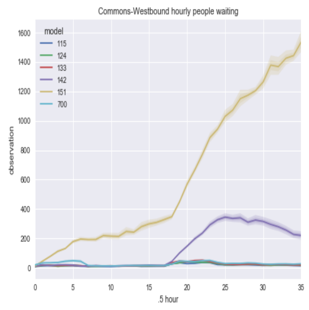
(c) Avg. Queue Length



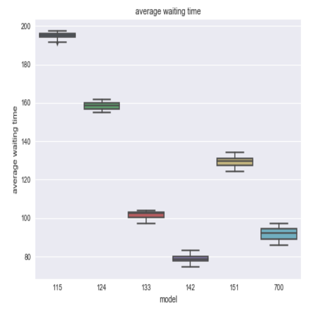
(d) Commons-Eastbound



(e) Commons-Westbound



(f) Avg. Waiting Time

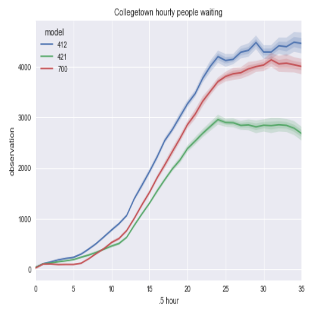


Looking at only graphs (a), (b), (d), and (f), it is difficult to tell which model performed the best overall. For each of them, it seems that a different model performed the best at each stop. The only discernible pattern is that the 142 and 700 models tended to be on the better end of all of the measured stops. This is corroborated by graphs (c) and (f); models 142 and 700 were the best two performers on queue length and wait time. Because of the higher variance in performance of queue length by model 700, the best overall performer in this subset seems to be model 142.

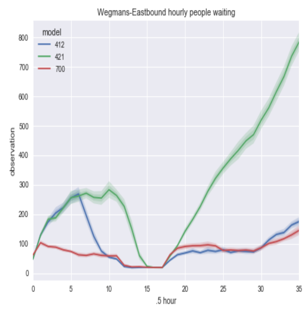
In addition, we tested models that fixed Route 1 with two, three, and four buses. The behavior of these models seemed to exist on a spectrum, with the one-fixed models being on one end and the four-fixed models being on the other end. For this reason, we are only showing the important extreme ends of the behavior spectrum. The models we show now are 421 and 412.

Table 3: Hourly Queue Length

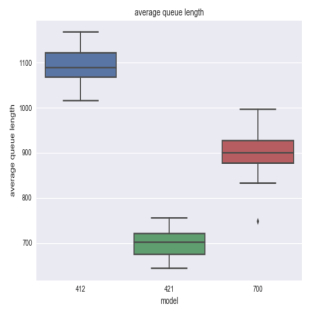
(a) Collegetown



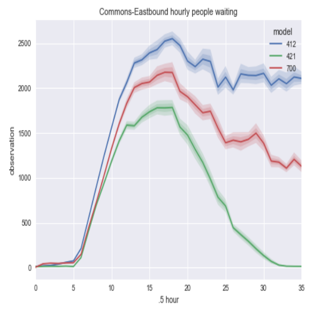
(b) Wegmans-Eastbound



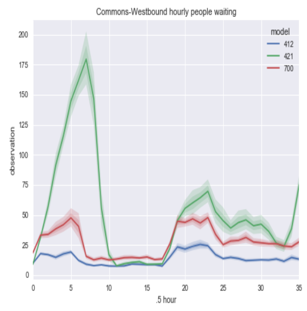
(c) Avg. Queue Length



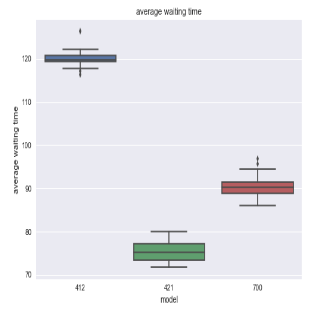
(d) Commons-Eastbound



(e) Commons-Westbound



(f) Avg. Waiting Time



Looking at Table 3 through a similar analytical lens as we did with Table 2, the 421 model seems to provide the best results, and it is our new leader. Based on graphs (a) and (d), model 421 seems to handle the highly-frequented Collegetown-Commons loop much better than either 412 or 700. We believe that this handling of the most high stakes route is what led to the model far outperforming the others in terms of mean queue length and mean waiting time. In fact, it seems to be so important that the 421 model's relatively poor performance on Wegmans-Eastbound and Commons-Westbound stops (shown in graphs (b) and (e)) seems not to affect our overall performance metrics very severely. In contrast with the one-fixed models from before, while performance improvements on the Collegetown-Commons loop were notable, fewer buses on

Route 1 caused poor performance on every other area of the map. For this reason, model 421 is a better candidate. However, the 700 performed well enough in comparison that it will continue to act as our baseline.

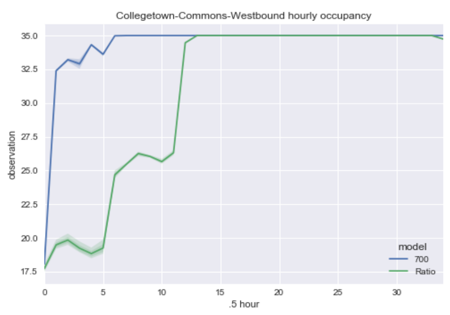
III. Dynamic Scheduling

Rate (#/hr)	6:00-9:00	9:00-12:00	12:00-15:00	15:00-18:00	18:00-21:00	21:00-24:00
Weg-Com	125	225	125	300	225	200
Weg-Ctown	325	175	100	175	225	300
Com-Ctown	175	850	400	126	225	200
Com-Weg	175	125	150	325	225	200
Ctown-Weg	325	150	125	250	225	300
Ctown-Com	150	500	850	600	225	150

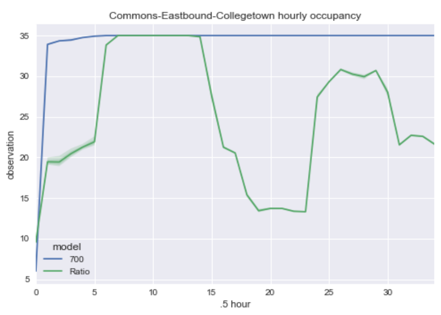
For our third simulation, we modified our models to adjust dynamic scheduling. Based on arrival data from the table above, we present the bus schedule in a list of tuples (x,y,z) where each item represents the number of buses on Route 1, Route 2, and Route 3 respectively. The list has a length of six because we assume a given bus will switch routes every three hours (the arrival rate period length) at each arrival period pivot point over the 18-hour-long operation time. We believe the system is inherently prone to congestion because of the randomness introduced by the non-homogeneous arrival rates.

Table 4: Unoccupied Routes

(a) Collegetown-Commons West



(b) Commons East-Collegetown



We first distributed the buses by the ratio of rates as follows:

- $[(3,2,2),(1,5,1),(1,5,1),(1,3,3),(2,2,3),(3,3,1)]$

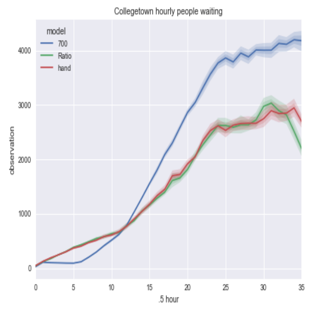
Then, we manually optimized the system by observing non-occupied bus routes at particular times and changing those routes toward more congested areas. Table 4 shows how effective this method was. On both long trip legs on the Collegetown-Commons loop, the ratio model had as many or fewer unoccupied routes as the 700 model. This means that buses were being repurposed effectively to meet the varying demands of riders across the map. From this visualization, we were also able to deduce that those unoccupied buses can change the route to more congested area – particularly re-routing the first, second the last interval to increase the buses in the Route3. Our manual optimization result is as follows:

- $[(2,2,3),(1,4,2),(1,5,1),(1,3,3),(2,2,3),(3,2,2)]$

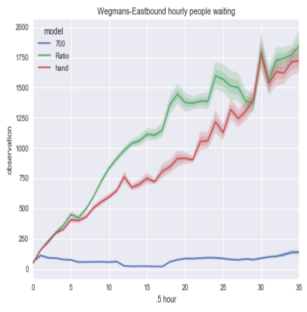
As evidenced in graphs (c) and (f) in Table 5, the manually optimized model seems to be the best so far, beating even the 421 model from before. It had an mean waiting time of 70 minutes and an mean queue length of 700. However, there is still clearly work to be done. As shown in graph (b) of table 5, the manually optimized model still runs up a large queue at the Wegmans-Eastbound stop when compared to our baseline 700 model.

Table 5: Hourly Queue Length

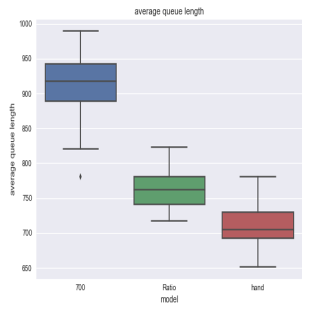
(a) Collegetown



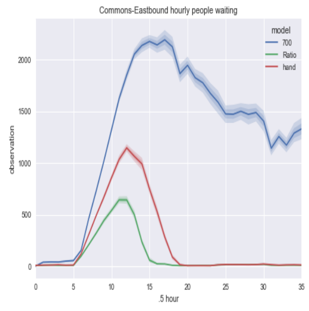
(b) Wegmans-Eastbound



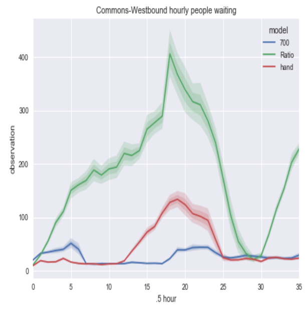
(c) Avg. Queue Length



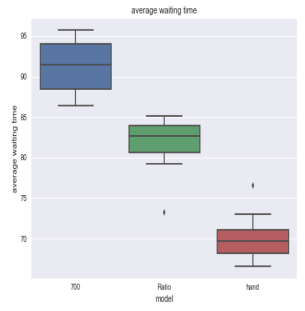
(d) Commons-Eastbound



(e) Commons-Westbound



(f) Avg. WAiting Time



IV. Bayesian Optimization

After manually observing the hourly utilization rate in the previous section, we decided to automate the parameter tuning process by applying the Bayesian optimization using the open source Python library *pysmac*. We first concluded that, based on our input observation, at least 1 bus has to be constantly running on the Route 1. Given this assessment, we defined the function with 36 inputs (6 bus routes schedule for the rest of 6 buses) and with the output, which is the metric we optimize. For each optimization iteration, we repeat 10 simulations for the given set of parameters and observe the sample mean as an output. In this simulation, we defined 3 metrics to minimize:

- Average waiting time
- Average queue length
- Number of people who wait 2+ hours (avg. time to show Hypothermia symptoms[1])

The optimization took about an hour for each metric with multi-core processing. We present our optimized bus scheduling in the format introduced in the previous section.

- $[(4,2,1),(3,3,1),(3,4,0),(5,1,1),(3,2,2),(5,1,1)]$ (Minimum avg. waiting time)
- $[(4,1,2),(3,3,1),(4,3,0),(5,1,1),(3,2,2),(5,1,1)]$ (Minimum avg. queue length)
- $[(5,1,1),(3,3,1),(4,3,0),(2,4,1),(3,1,3),(5,1,1)]$ (Lowest Hypothermia rate)

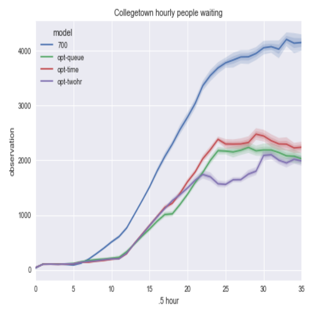
Interestingly enough, all 3 optimization tasks resulted in fairly similar re-rerouting schedules. All of the suggested schedules respond well with the "rush hour" we observe in the 2nd and 3rd intervals by increasing the number of buses on Route 2. Also the algorithm was able to detect another rush hour between collegetown and wegmans in both the first and last interval. As we see in Figure 6, all of the optimized models performed extremely well and beat the benchmark by a significant margin with an mean queue length of 450-500, and waiting time of 50-60 min.

[1] I made this statistic up - it may or may not be accurate.

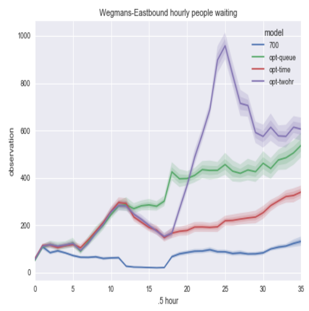
Table 6: Hourly Queue Length

opt-queue, opt-time, opt-twohr refer to the models with lowest queue length, waiting time and hypothermia rate respectively.

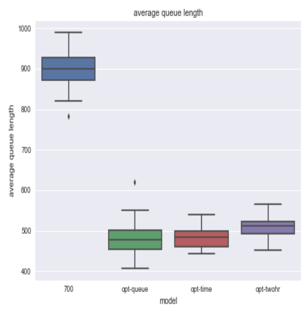
(a) Collegetown



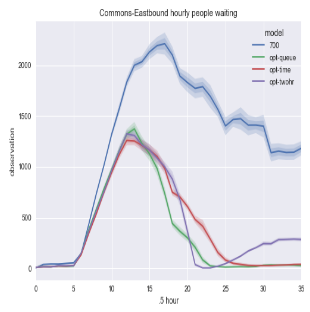
(b) Wegmans-Eastbound



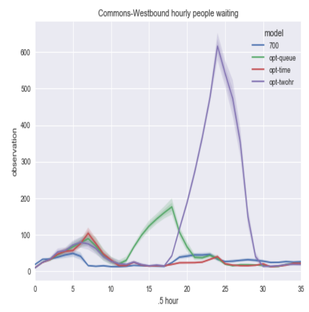
(c) Avg. Queue Length



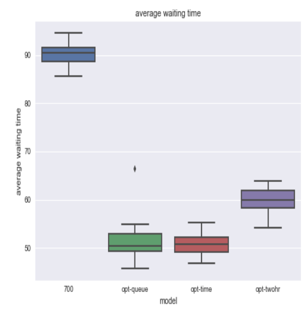
(d) Commons-Eastbound



(e) Commons-Westbound



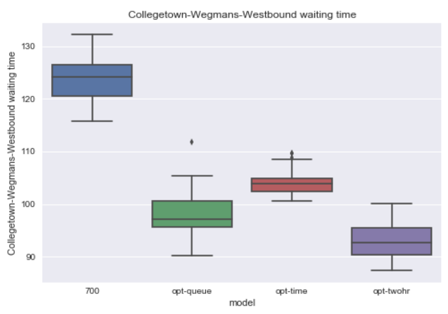
(f) Avg. Waiting Time



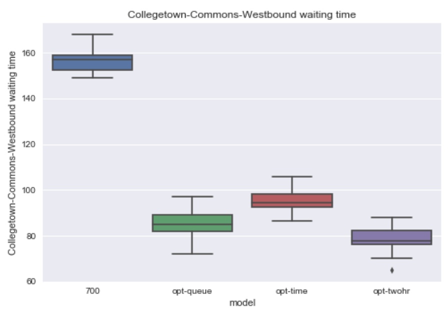
One striking difference is in the 4th interval where the model with the lowest hypothermia rate continues sending the buses to the Route 2. During the 4th interval, the amount of travel requests between commons and wegmans increase significantly. However, the model decides to prioritize people in collegetown, who have been waiting for the bus, instead of answering the recently arrived passengers in commons/wegmans. This is the reason why we observed 2 peaks in the time-series and a small dip in collegetown. As we see in Figure 7, the model improves the waiting time in collegetown for both of destinations.

Table 7: Waiting Time at Busiest Stops

(a) Collegetown-Wegmans



(b) Collegetown-Commons



VII. Conclusions

Given the results, we propose two re-routing strategies:

- $[(4,1,2),(3,3,1),(4,3,0),(5,1,1),(3,2,2),(5,1,1)]$
- $[(5,1,1),(3,3,1),(4,3,0),(2,4,1),(3,1,3),(5,1,1)]$

The first strategy demonstrated the smallest sample mean for both queue length (450-500) and waiting time (48-54) among all models we tested. This strategy was found through Bayesian optimization to minimize the mean queue length, which resulted in almost identical result with the shortest waiting time strategy. We also propose the second strategy which lowers the number of passengers with extreme waiting time (defined as 2 hours or longer). Even though the mean values of waiting time and queue length are larger than the previous model, lowering the number of extremes will improve the customer experience.

Finally, the extreme versatility of our methods allows us to further investigate different metrics, parameters to discover other strategies tailored towards specific cases. The potential improvements are as follows:

- Consider delay strategies for each bus
- Allow more frequent route-changing
- Optimize with harmonic average instead of mean to avoid extremes
- Consider abandonment of passengers

VIII. Appendix

Our Python implementation of a general transportation system consists of the following classes (some methods and attributes are not listed):

Event Represents the arrival/departure of a bus to/from a bus stop.

Attributes:
- `time`: Time at which this discrete event occurred.
- `type`: Either "arrival" or "departure".
- `bus`: Bus object to which this event applies.
- `bus_stop`: BusStop object at which this arrival/departure event occurs.

Map Represents and controls all buses, bus stops, people within the modelled system.

Attributes:
- `routes`: A list of all routes that a bus can travel on.
- `buses`: A list of all buses in the modelled system.
- `bus_stops`: A list of all bus stops in the modelled system.
- `event_queue`: Records all events occurring in the system; processed in chronological order.

Methods:
- `simulate(max_time)`: Simulates (i.e. processes events) the given map for `max_time` minutes.
- `collect_stats()`: Return summary statistics to a DataFrame.

Bus Represents a bus within the transportation system.

Attributes:
- `route`: Current route of the bus.
- `schedule`: Schedule of the bus, specified as the route for every 3-hour period.
- `next_stop`: Next stop that this bus should travel to.
- `passengers`: List of Person objects who are currently on the bus.
- `occupancy`: Current occupancy of the bus.
- `max_cap`: Maximum capacity of the bus (default=35).

Methods:
- `goes_to(stop)`: Whether this bus' current route contains a given bus stop.
- `request_route_change(route)`: Request a route change; may not be executed immediately.
- `execute_route_change()`: Execute a route change when conditions to do so are satisfied.
- `board()`: Allow all passengers who want to and will fit to board the bus.
- `arrive(stop)`: Board, depart, execute route change as necessary; generate departure event.
- `depart(stop)`: Calculate travel time, distance; generate subsequent arrival event.

BusStop Represents a bus stop within the transportation system.

Attributes:
- `num_waiting`: Number of people currently waiting at this stop.
- `people_waiting`: List of Person objects representing current people waiting.

Methods:
- `arrival(time)`: Simulate the arrival of a single person to this stop at `time`.
- `update(time)`: Simulate all arrivals to this stop up until `time`.

Person Represents a person within the transportation system.

Attributes:

- `origin`: BusStop object at which this person should board.
- `destination`: BusStop object at which this person should depart.

Route Represents a bus route within the transportation system.

Attributes:

- `stops`: List of BusStop objects that this route travels between.
- `distances`: Distances between each stop on this route.
- `switch_points`: Information required to transition between routes.
- `num`: Route number as specified by the TDOG.

For further details, see `https://github.com/yujiakimoto/PySimio`