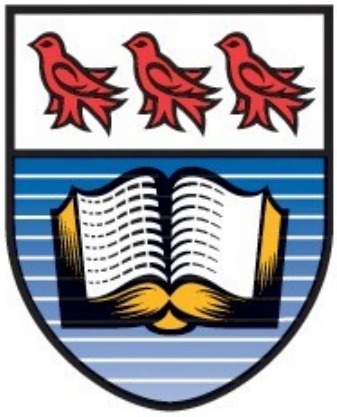
University of Victoria
Department of Computer Science

Simulation Study on Adaptive Traffic Light Control

In partial fulfillment
of the requirements of the CSC 446 Final Project

Fall 2024



By
Sebastien Perre and Henry Gross-Hellsen
Performed at: University of Victoria, Victoria

Contents

Abstract i

1 Problem Description 1

2 Data Collection 2
- 2.1 Bounding Box . . . 2
- 2.2 Overpass Turbo with Open Street Maps . . . 2
- 2.3 TomTom API . . . 3
- 2.4 Getting the Lanes Visually . . . 3
- 2.5 Getting the intersection length . . . 4

3 Simulation Parameters 5
- 3.1 Estimating the Arrival Rate . . . 5
- 3.2 Estimating the Processing Rate . . . 5
- 3.3 Number of Lanes . . . 5
- 3.4 Assumptions about the parameters . . . 6
  - 3.4.1 Arrival Rate . . . 6
  - 3.4.2 Processing rate . . . 6

4 Simulation Model: Fixed Control 7
- 4.1 Assumptions . . . 7
- 4.2 How the Simulation Works . . . 8
  - 4.2.1 Setup . . . 8
  - 4.2.2 Simulation . . . 8
- 4.3 Improvements . . . 8

5 Simulation Model: Adaptive Control 9
- 5.1 Reinforcement Learning Model . . . 9
- 5.2 Algorithmic Model . . . 9
  - 5.2.1 Simulation . . . 9
- 5.3 Improvements . . . 9

6 Analysis: Comparing Fixed and Adaptive Controls 11
- 6.1 Scenario Analysis . . . 11
- 6.2 Performance Analysis . . . 12

7 Conclusion 14

References **15**

Appendix **16**

Code Setup 16

API Queries 16

Overpass API 16

TomTom API 16

Abstract

Traffic congestion is a challenge among major cities. It contributes to air pollution, travel delays, fuel consumption, and many other issues. Health Canada estimates that traffic-related air pollution contributes to 1,200 premature deaths, 210,000 days of asthma symptoms, and 2,700,000 acute respiratory days accumulating at a socioeconomic cost of $9.5 billon [4]. Thus, for the sake of reducing traffic time and idle vehicle use, it is of paramount importance to create traffic lights that adapt to real-time traffic conditions. We collect real-time data to estimate traffic conditions to look at possible solutions to see how much better an adaptive control traffic light is compared to a fixed control traffic light. Our findings conclude an adaptive model improves traffic flow at a 5% level of significance with a few caveats.

All code can be found at https://github.com/sebperre/traffic-light-simulation

1 Problem Description

Traffic congestion is a challenge facing all over the world that contributes to air pollution, travel delays, fuel consumption, and many other issues. In Canada alone, Health Canada estimates traffic-related air pollution contributes to 1,200 premature deaths, 210,000 asthma symptom days, and 2,700,000 acute respiratory days accumulating at a socioeconomic cost of $9.5 billon [4]. Furthermore, this issue is more prevalent in marginalized communities as they are exposed to higher levels of traffic-related air pollution than the general population [5].

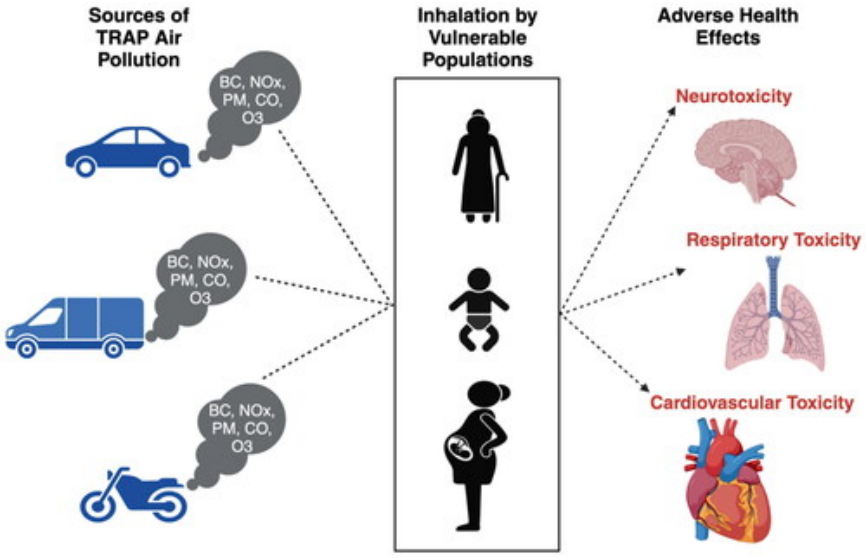


Figure 1: Some of the few effects that traffic-related air pollution causes. Taken from [5]

.

Our goal is to gather real-time data to estimate congestion and produce a model of a traffic-light based on a real street. We then use this data to produce a simulation of a fixed control traffic light and an adaptive traffic light to see if we can produce a better performing model for the adaptive traffic light control than the fixed traffic light control.

All code can be found at https://github.com/sebperre/traffic-light-simulation

2 Data Collection

The data collection process consists of using the Overpass Turbo with OSM (Open Street Maps) API to gather the longitudes and latitudes of the traffic lights to then pass into the TomTom API which gives us real-time traffic data. As the TomTom API gives us real-time traffic data, we set up a Linux cron job to ping the API with various traffic lights. To get the lanes and intersection length, we use Google Maps to count the number of in and out-lanes and use their mesaure tool.

2.1 Bounding Box

To search for traffic lights using the Overpass Turbo with OSM API we have to get a bounding box so the API can search for traffic lights. A free online tool is the BoundingBox website by Klokantech which provides an interactive way to produce a bounding box on the map.

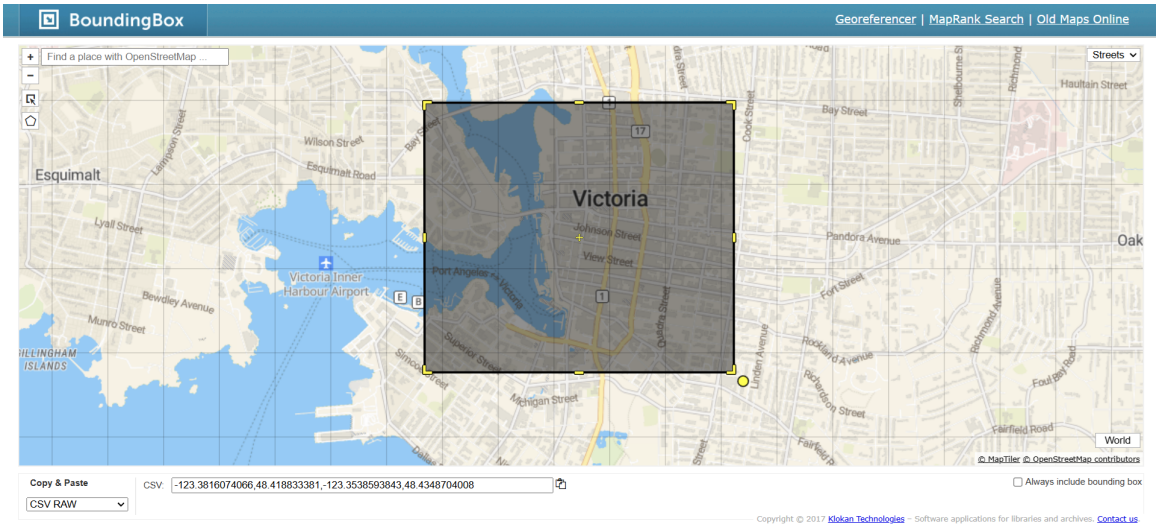


Figure 2: Bounding Box tool displaying geographic coordinates in CSV RAW format, represented by the bottom-left (longitude, latitude) and top-right (longitude, latitude) points. Taken from [6] and click here to access the tool.

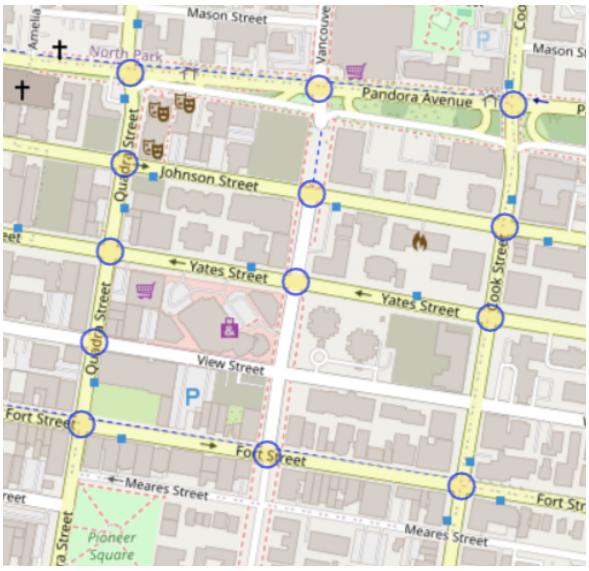
As shown in Figure 2, the BoundingBox tool provides geographic coordinates. Note that the format is in the form bottom-left point (longitude, latitude) and top-right point (longitude, latitude). We pass these points to the Overpass Turbo with OSM API.

2.2 Overpass Turbo with Open Street Maps

The Overpass Turbo with OSM API provides a way to retrieve traffic lights in a bounding box, using input coordinates. Note you can use this website [7] to test what you get from the API. Furthermore, you must enter the coordinates as latitude (bottom-left), longitude (bottom-left), latitude (top-right), longitude (top-right) (Note this

```
{
  "type": "node",
  "id": 25832954,
  "lat": 48.4323992,
  "lon": -123.3209411,
  "tags": {
    "highway": "traffic_signals"
  }
},
{
  "type": "node",
  "id": 25832995,
  "lat": 48.4321670,
  "lon": -123.3224050,
  "tags": {
    "highway": "traffic_signals"
  }
},
```


(a) Data from the API in JSON



(b) Traffic lights covered by bounding box

Figure 3: Example of using the Overpass Turbo with OSM API to gather streetlight coordinates. Taken from [7] and click here to access the website

differs from what BoundingBox gives you). Finally, you can see an example of a query by going to the Overpass API in the Appendix.

2.3 TomTom API

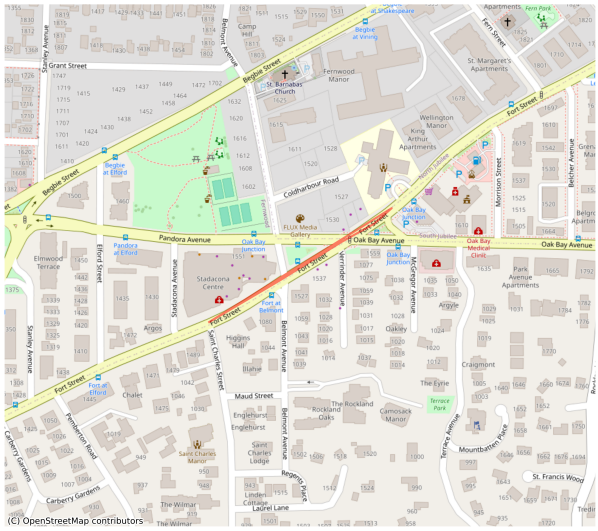
After gathering the longitudes and latitudes of the street lights, we use the TomTom API to gather real-time traffic data of the traffic light. As shown in Figure 4, we get the traffic data for the road(s) around the traffic light, but this will be a good enough approximation to estimate the traffic flow of the intersection. We get the current speed (kmph), free flow speed (kmph), current travel time (seconds), and free flow travel time (seconds) which we will use to estimate our parameters. As the TomTom API only provides us with real-time traffic data, we set up a cronjob everyday for hour 0(12AM), 6(6AM), 9(9AM), 12(12PM), 15(3PM), 18(6PM), 21(9PM) PST to get how the traffic is for morning, rush hour and evenings. Finally, the information for the API was gathered from here [8] and an example of a TomTom API and the cronjob setup can be found by going to TomTom API in the appendix.

2.4 Getting the Lanes Visually

Now, we need to figure out how many queues we need to simulate the traffic lights. There is no way easy to get the lanes programmatically so we retrieve them manually by using google maps and manually counting the in-lanes and out-lanes.

```
"flowSegmentData": {
    "frc": "FRC4",
    "currentSpeed": 24,
    "freeFlowSpeed": 42,
    "currentTravelTime": 132,
    "freeFlowTravelTime": 75,
    "confidence": 0.99,
    "roadClosure": false,
    "coordinates": {
        "coordinate": [
            {
                "latitude": 48.42644437088044,
                "longitude": -123.3227241795338
            },
            {
                "latitude": 48.426455105675196,
                "longitude": -123.3222346765022
            },
```


(a) Data from the API in JSON



(b) Traffic Data Location

Figure 4: Example of using the TomTom API on the Traffic Light for Pandora Avenue and Oak Bay Avenue.

2.5 Getting the intersection length

We use google maps to gather the distance of the intersection by using the measure tool available.

Figure 5: Example of counting the in-lanes and out-lanes and getting the distance of the intersection.

.

3 Simulation Parameters

There are three simulation parameters we need for modeling the intersection, the number of in-lanes for each direction, the arrival rate and the processing rate.

3.1 Estimating the Arrival Rate

We use freeflow speed (kmph), current speed (kmph) and number of in-lanes to estimate the arrival rate. The formula was derived as

$$\lambda = N_{\text{in-lanes}} \cdot \lambda_{\text{Free Flow per Lane}} \cdot R$$

where $N_{\text{in-lanes}}$ is the number of in-lanes, $\lambda_{\text{Free Flow per Lane}}$ is the arrival rate of a lane in free flow and R is the congestion factor where R is

$$R = \frac{s_{\text{Free}}}{s_{\text{Current}}}$$

and $\lambda_{\text{Free Flow per Lane}}$ can be estimated as,

$$\lambda_{\text{Free Flow per Lane}} = \frac{\frac{s_{\text{Free}}}{3.6}}{c + L_{\text{Car}}} = \frac{s_{\text{Free}}}{3.6 \cdot (c + L_{\text{Car}})}$$

where s_{Free} is the free flow speed, c is the distance between cars and L_{car} is the average length of a car. We can use 4.48 meters as L_{car} [2]. We will also use double the car length as c advised by Broker Link [1] so we have $c = 8.96$ meters. As we want to model congestion we will ignore the case for $\lambda < \lambda_{\text{Free Flow per Lane}}$.

3.2 Estimating the Processing Rate

The processing rate is the number of cars that can be processed in a time unit from the start of the intersection to the end of the intersection. We will calculate it as,

$$\mu = \frac{\frac{s_{\text{Free}}}{3.6}}{d} = \frac{s_{\text{Free}}}{3.6 \cdot d}$$

where d (meters) is the distance of the intersection and s_{Free} is the free flow speed.

3.3 Number of Lanes

We retrieve the number of in-lanes and out-lanes by manually counting them. Please check Section 2.4 for more context.

3.4 Assumptions about the parameters

We made multiple simplifying assumptions for estimating the parameters that affect the simulation.

3.4.1 Arrival Rate

Based on our formula above, we assume the arrival rate for each lane is constant, but it could very well be that certain-lanes are more congested than others. Furthermore, we assume that every car goes at the speed limit, not above or below. Finally, we assume the cars are separated by double the car length which could very well not be the case.

3.4.2 Processing rate

The processing rate assumes no acceleration. We added a random delay to accommodate for a start-from-stop scenario.

4 Simulation Model: Fixed Control

We make our simulation only have 4 directions with any number of lanes, but this code can be extended to have different directions and intersections.

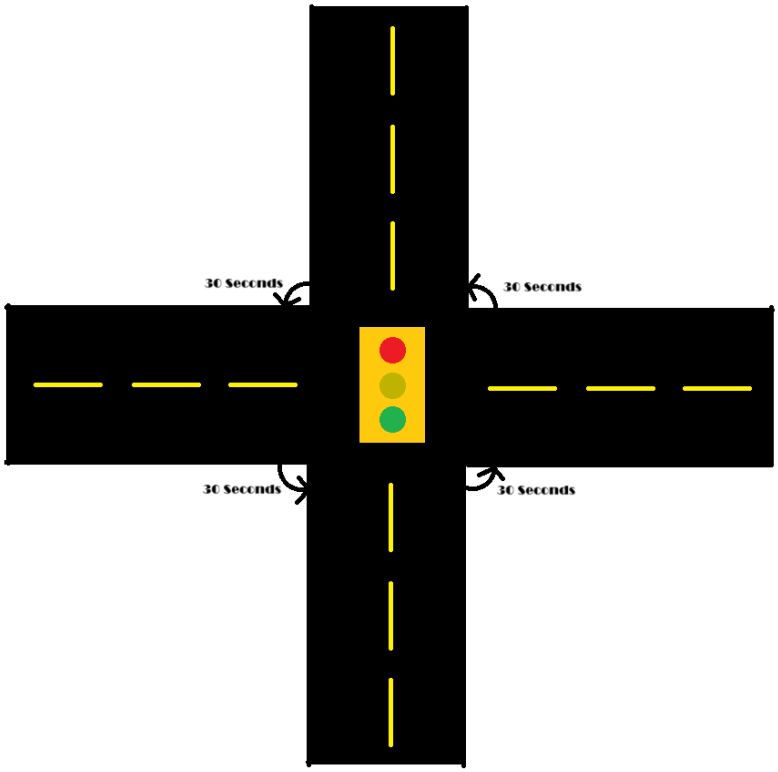


Figure 6: A visual for the fixed light traffic light

.

4.1 Assumptions

We do not address left turns.
We do not address possibility of accidents.
We do not address possibility of traffic after our intersection.
We do not address illegal behavior.
We assume that each vehicle is identical, and follow the speed limit precisely.

4.2 How the Simulation Works

4.2.1 Setup

We start off by reading our intersection data which was manually retrieved via the process mentioned in 2.4 and 2.5. Based on this data, we generate queues for each of the lanes and get the current speed, free flow speed, current travel time and free flow travel time. We use the TomTom data to generate our parameters lambda free flow, congestion factor, lambda and vehicle processing rate (These formulas are shown in 3.1 and 3.2 respectively). We then pass in a parameter to determine how we should split up the lambdas lane by lane. Finally, we simulate all the interarrival times for the lanes to get car arrival time and do some processing on the interarrival times.

4.2.2 Simulation

We add the arrival times to a priority queue with its arrival time, Id and lane. Then, we start the traffic light with a random direction (either North-South or East-West) and enter the cars in the priority queue. Whenever a light turns from red to green, we add a delay to the first car to compensate for it taking time to speed up. Direction contractions represent a grouping of opposite lanes, where we have one queue per lane entering the intersection. Thus, we have four groups (one per approach), where each group has two queues representing two lanes entering the intersection. The output lanes are not considered, as we assume that a vehicle does not encounter traffic or other unexpected conditions once leaving the intersection. For traffic light control, we have the light on a fixed 30-second interval. I.e. North-South is processed for 30 seconds, followed by East-West. When the current queue group is empty, and we have enough time to process a right on red, we process a right on red (rightmost lanes of groups with a red light) as done similarly to above.

4.3 Improvements

We would like to have per-vehicle attributes, ie. priority, weight, speed, etc., but we do not have all the necessary data to implement this. Implementation of per-vehicle attributes would allow the following:

- Vehicle headway: Would be calculated based on the vehicle length, speed, and distance from the previous vehicle.
- Vehicle priority: Would allow prioritization of certain vehicles.

Implement Canada/BC mandated clearing of lanes for emergency vehicles (even though there may already be a clear lane)

5 Simulation Model: Adaptive Control

We make our simulation only have 4 directions with any number of lanes, but this code can be extended to have different directions and intersections.

5.1 Reinforcement Learning Model

We tried using a reinforcement learning model for an adaptive model, but it didn't do well and we decided to scrap it. We used the following variables to train it on (We don't include hyperparameters like γ in this table),

Variables	Values	Amount
States	Number of cars and lanes in north, south, east and west direction and current light direction (north and south or east and west)	9
Actions	Turn north and south lights green (turn east and west lights red) and turn east and west lights green (turn north and south lights red)	2
Awards	+1 for each car pass through the intersection and $-0.5(e^{x/50} - 1)$ for each car waiting in a red light lane	2
Model	Deep Q-Network	Not Applicable

Table 1: Reinforcement Learning Variables

After this failed attempt, we decided to come up with an algorithmic way to determine when to turn the traffic lights green.

5.2 Algorithmic Model

The sections for this ADAPTIVE/DYNAMIC control model are identical to FIXED (refer to 4), with differences as follows.

5.2.1 Simulation

For DYNAMIC traffic light control, each queue group is processed until empty, and is then switched to the next group.

5.3 Improvements

Currently, we process a queue (lane) until it is empty. However, it would be more ideal to process each queue up to a certain length of time, dependent on the length of the other queues. A side effect of having an algorithmic control is that we do not have enough time to process right turns on red. A solution to this would be to implement a system where a vehicle turning right does NOT wait for an appropriate opening, and instead forces it's way through resulting in other right-of-way vehicles to slow down in order to avoid a collision. This would closely

model common (but illicit) real-life behavior, and was considered out-of-scope as we do not consider illegal behavior in our model.

6 Analysis: Comparing Fixed and Adaptive Controls

6.1 Scenario Analysis

We compare our two models with 1000 drivers for December 1st at 9AM (Rush Hour), 12PM (Lunch), 3PM (End of School Day), 6PM (End of Work Day). We test this on the intersection at Oak Bay Avenue and Foul Bay Road in Victoria. Note that we run this simulation 10 times and get the average for each,

Table 2: Metrics for Fixed and Dynamic Light Control Over 10 Runs (We use all lanes equally to populate with vehicles)

Time	Control	Avg. Waiting Time (s)	Throughput (veh/min)	CI (a=0.05)	Max Queue Length	Efficiency (s)
09:00	Fixed	209.34	109.51	[105.89, 113.13]	118.10	548.99
	Dynamic	215.93	107.28	[104.60, 109.96]	121.72	564.20
12:00	Fixed	210.74	107.79	[104.16, 111.42]	118.75	559.49
	Dynamic	217.15	107.72	[104.91, 110.53]	123.19	563.18
15:00	Fixed	212.21	107.74	[104.37, 111.10]	118.47	560.64
	Dynamic	221.35	107.62	[104.47, 110.78]	123.98	566.23
18:00	Fixed	212.75	107.21	[104.50, 109.92]	118.58	562.96
	Dynamic	224.36	107.02	[103.89, 110.14]	124.59	570.47
21:00	Fixed	211.97	106.64	[104.13, 109.15]	119.32	566.36
	Dynamic	225.20	107.68	[104.34, 111.01]	125.17	569.04

Table 3: Metrics for Fixed and Dynamic Light Control Over 10 Runs (1 Lane 1 Direction)

Time	Control	Avg. Waiting Time (s)	Throughput (veh/min)	CI (a=0.05)	Max Queue Length	Efficiency (s)
09:00	Fixed	1707.92	17.13	[16.63, 17.63]	975.90	3507.86
	Dynamic	1611.18	19.09	[17.55, 20.62]	972.73	3300.82
12:00	Fixed	1710.97	17.15	[16.85, 17.45]	977.10	3503.36
	Dynamic	1513.47	21.38	[19.16, 23.60]	969.33	3104.92
15:00	Fixed	1715.54	17.11	[16.88, 17.34]	975.83	3511.64
	Dynamic	1444.57	22.74	[20.54, 24.94]	967.43	2966.78
18:00	Fixed	1721.22	17.05	[16.84, 17.26]	975.53	3524.11
	Dynamic	1393.88	23.85	[21.65, 26.06]	965.74	2865.09
21:00	Fixed	1714.78	17.12	[16.94, 17.31]	975.26	3508.92
	Dynamic	1348.54	24.83	[22.66, 27.00]	964.18	2774.31

The data in Table 2 shows when barraged with cars at all four lanes, both models are comparable at a significance level of 5%. This makes sense as if all lanes are backed up, the dynamic model won't perform better than the fixed model.

Table 4: Metrics for Fixed and Dynamic Light Control Over 10 Runs (2 Lanes 1 Direction)

Time	Control	Avg. Waiting Time (s)	Throughput (veh/min)	CI (a=0.05)	Max Queue Length	Efficiency (s)
09:00	Fixed	567.70	33.51	[32.09, 34.92]	468.40	1796.08
	Dynamic	536.08	39.27	[35.28, 43.26]	472.55	1653.13
12:00	Fixed	548.33	34.18	[33.29, 35.06]	470.75	1760.77
	Dynamic	528.65	42.32	[38.15, 46.48]	471.96	1568.95
15:00	Fixed	550.08	34.31	[33.64, 34.98]	471.13	1753.17
	Dynamic	521.83	44.80	[40.54, 49.07]	471.08	1504.55
18:00	Fixed	552.95	33.95	[33.35, 34.55]	473.33	1772.46
	Dynamic	505.18	48.49	[43.85, 53.13]	469.73	1432.12
21:00	Fixed	549.88	33.87	[33.29, 34.46]	474.44	1777.70
	Dynamic	499.45	50.18	[45.73, 54.62]	469.15	1388.12

Table 5: Metrics for Fixed and Dynamic Light Control Over 10 Runs (1/10th normal Lambda, all lanes equally populated with vehicles)

Time	Control	Avg. Waiting Time (s)	Throughput (veh/min)	CI (a=0.05)	Max Queue Length	Efficiency (s)
09:00	Fixed	16.66	69.75	[66.18, 73.32]	13.60	864.26
	Dynamic	19.20	69.87	[68.74, 71.01]	16.93	862.10
12:00	Fixed	16.76	69.92	[67.60, 72.24]	14.40	862.27
	Dynamic	21.38	70.18	[69.13, 71.23]	18.77	858.33
15:00	Fixed	16.57	69.98	[68.28, 71.69]	14.63	860.95
	Dynamic	22.31	69.81	[68.82, 70.80]	20.04	862.93
18:00	Fixed	16.16	69.52	[68.09, 70.95]	14.20	866.60
	Dynamic	23.70	69.91	[68.99, 70.83]	21.07	861.68
21:00	Fixed	16.09	69.69	[68.47, 70.92]	14.32	864.17
	Dynamic	25.59	69.80	[68.92, 70.67]	22.25	863.13

For table 3 when you have cars in 1 lane and in 1 direction, the dynamic model performs significantly better at 5% (Throughput) for 12:00, 15:00, 18:00 and 21:00. Furthermore, the dynamic model has better efficiency. For table 4 we see similar results for 3, but the difference is bigger.

6.2 Performance Analysis

For equally populated lanes ($\tilde{1}25$ per lane, $\tilde{2}50$ per approach, 1000 total), the metrics were approximately the same. However, average waiting time and efficiency suffers for the dynamic model in this scenario. This was expected, due to our dynamic algorithm. (See 5.2.1 and 5.3 for improvements). This applies for Tables 2 and 5. For all other scenarios, we saw a general trend where the dynamic model had lower average waiting time, higher throughput, smaller queue length, and a lesser efficiency/total run-time.

Table 6: Confidence Intervals for CRN of Throughputs and Average Waiting Times (100 Runs for each Time. We use all lanes equally to populate with vehicles)

Time	CRN of Throughputs	Average Waiting Time
09:00	(-0.2272, -0.0337)	(-30.3953, -5.1302)
12:00	(-0.1773, -0.0046)	(-36.6591, -11.7853)
15:00	(-0.2298, -0.0590)	(-25.6041, -1.1015)
18:00	(-0.2121, -0.0157)	(-29.8231, -3.9752)
21:00	(-0.1967, -0.0155)	(-34.5252, -8.8391)

Table 7: Confidence Intervals for CRN of Throughputs and Average Waiting Times (100 Runs for each Time. 1 Lanes 1 Direction)

Time	CRN of Throughputs	Average Waiting Time
09:00	(-0.2993, -0.2268)	(696.873, 817.941)
12:00	(-0.3474, -0.2669)	(743.652, 870.781)
15:00	(-0.3089, -0.2386)	(734.831, 856.823)
18:00	(-0.2754, -0.2102)	(674.210, 791.627)
21:00	(-0.3052, -0.2324)	(714.750, 826.985)

Table 8: Confidence Intervals for CRN of Throughputs and Average Waiting Times (100 Runs for each Time. 2 Lanes 1 Direction)

Time	CRN of Throughputs	Average Waiting Time
09:00	(-0.5920, -0.4636)	(75.020, 131.516)
12:00	(-0.5523, -0.4271)	(46.693, 107.353)
15:00	(-0.5380, -0.4046)	(25.652, 89.489)
18:00	(-0.5516, -0.4162)	(41.036, 101.905)
21:00	(-0.5395, -0.4039)	(30.882, 95.103)

The reason waiting time increases is due to our prioritization of maximizing throughput. A side effect is some vehicles may wait for an unreasonable amount of time since there are more vehicles going one way than the other. A possible fix in a future implementation would be to switch the light every certain amount of seconds when there are cars present in the lane which has a red light.

7 Conclusion

The dynamic model overall performs well when vehicles are unequally distributed among queues, such as all vehicles being on 1 lane (Table 3, trivial), or all vehicles in 2 lanes (Table 4); where average efficiency, throughput, and waiting time are less, but queue lengths increase. It performs equally or worse in all metrics for scenarios with evenly distributed vehicles (Table 2, Table 5).

In conclusion, we confirm that our dynamic model outperforms the fixed model in most scenarios, but fails to do so in scenarios where all lanes are equally filled with vehicles.

Solutions to scenarios where the dynamic model under-performs would be to implement a time parameter, such that each queue (lane) would not wait for too long. This would depend on time spent in the queue for each vehicle. Another solution would be to vary depending on the specific states of the queues. This would depend on the queue lengths. These suggestions would necessitate implementations of improvements from 4.3 and 5.3.

References

[1] BrokerLink. *What is a safe distance between cars when driving?* Accessed December 10, 2024. Feb. 2024. URL: https://www.brokerlink.ca/blog/what-is-a-safe-distance-between-cars-when-driving (visited on 12/10/2024).

[2] Cartelligent. *What is the average car length?* Accessed December 10, 2024. URL: https://cartelligent.com/average-car-length (visited on 12/10/2024).

[3] BrokerLink Communications. *Traffic light controls*. Accessed on December 8th, 2024. June 2024. URL: https://www.brokerlink.ca/blog/traffic-light-controls.

[4] Health Canada. *Traffic-Related Air Pollution*. Accessed on December 10, 2024. n.d. URL: https://www.canada.ca/en/health-canada/services/air-quality/outdoor-pollution-health/traffic-related.html (visited on 12/10/2024).

[5] Rahanna N. Khan et al. "Traffic-related air pollution in marginalized neighborhoods: a community perspective". In: *Inhalation Toxicology* 36.5 (2024). PMID: 38618680, pp. 343–354. DOI: 10.1080/08958378.2024.2331259. eprint: https://doi.org/10.1080/08958378.2024.2331259. URL: https://doi.org/10.1080/08958378.2024.2331259.

[6] Klokan Technologies. *Bounding Box Tool: Metadata Enrichment for Catalogue Records by Visually Selecting Geographic Coordinates (Latitude / Longitude) for Maps*. Accessed: December 8, 2024. 2017. URL: https://boundingbox.klokantech.com/.

[7] Martin Raifer. *Overpass Turbo*. Accessed: December 8, 2024. 2012. URL: https://overpass-turbo.eu/.

[8] TomTom. *Traffic API Documentation: Traffic Flow - Flow Segment Data*. Accessed: December 8, 2024. TomTom International BV. 2024. URL: https://developer.tomtom.com/traffic-api/documentation/traffic-flow/flow-segment-data.

Appendix

Code Setup

API Queries

Overpass API

```
# Example Arguments
AREA_TO_SEARCH = "48.403381,-123.395804,48.447075,-123.313201"
TRAFFIC_STOP_AMOUNTS = 10
query = f"""
        [out:json][timeout:25];
        node["highway"="traffic_signals"]({AREA_TO_SEARCH});
        out body {TRAFFIC_STOP_AMOUNTS};
        >;
        out skel qt;
        """
```

Listing 1: Query to get traffic lights

TomTom API

```
# Example Arguments
SECRET_KEY = "a1b2c3d4e5f6g7h8i9j0k12345" # Get the query key from TomTom
# The traffic_light is from the Overpass API
url_tomtom =
   f"https://api.tomtom.com/traffic/services/4/flowSegmentData/absolute/22"
    + "/json?key={SECRET_KEY}&point={traffic_light["lat"]},"
    + "{traffic_light["lon"]}&thickness=1"
```

Listing 2: Query to get TomTom Data

```
0 6,9,12,15,18,21,0 * * * ~/workspace/main.py >> ~/workspace/logfile.log 2>&1
```

Listing 3: Cronjob Setup