Congestion-Aware Routing for Multi-Class Mobility-on-Demand Service

Semester 8 Project Thesis

By: Niharika Shrivastava (IIT2016501)
Supervisor: Prof. Malika Meghjani
Co-Supervisor: Prof. O.P. Vyas

Department of Information Technology
Indian Institute of Information Technology
Allahabad, India

July 5, 2020

Candidate's Declaration

I declare that the project work titled "Congestion-Aware Routing for Multi-Class Mobility-on-Demand Service" done at the Singapore University of Technology and Design, Singapore and submitted at the Indian Institute of Information Technology, Allahabad is the bonafide work of Niharika Shrivastava(IIT2016501). It is a genuine record of my study carried out from January 2020 till June 2020 under the guidance of Prof. Malika Meghjani and Prof O.P. Vyas. Due acknowledgements have been made in the text to all the materials used.

Malika

.....

Prof. Malika Meghjani

(Supervisor)

Date: ..../..../.....

Place: Allahabad

.....

Prof. O.P Vyas

(Co-Supervisor)

Abstract

Recent developments in urban mobility have demonstrated the potential capacity of road networks to accommodate increasing traffic demand. However, a majority of these approaches focus on sub-optimal user-centric greedy solutions which require expensive real-time traffic information. This project presents a near-optimal and computationally efficient customer routing algorithm in a congested network. First, our aim is to provide congestion-aware routes for mobility-on-demand services that operate using a combination of a heterogeneous multi-class fleet. Every customer trip is distributed into multiple legs, i.e., walking or cycling for the first and last leg of the trip; cars or public transport for the middle leg of the trip. Second, we provide optimal transit points from one leg to another for this intermodal (multi-class) service, such that the overall travel time is at least within a capacity-bound transportation network. Finally, we build this framework into a social model by fulfilling all trip demands in a system-optimal way. We showcase the effectiveness of our framework by using Singapore's network data from OpenStreetMap coupled with its real-time traffic speed-band data. We achieve 74.26% increase in network utilization for 500 customers for different peak times along with a 0% congestion rate by using a multi-class setup. Our proposed algorithm to find optimal transit points within trips is 70.1% more computationally efficient with an insignificant average delay of 4 seconds compared to an exhaustive search. Moreover, our solution also recommends alternate paths that are user-optimal.

Contents

1 Introduction

1.1 Motivation

Traffic congestion on urban road networks has become increasingly problematic since the 1950s [1]. With an increase in vehicles on the road and capacity of the roads remaining approximately constant, the speed of the traffic stream slows [2]. This results in higher travel times and increased pollution. With the increasing popularity of on-demand mobility services [3, 4, 5] like Uber, Grab, Yulu and Lyft, the traffic demand has started to reach road capacities more often, thereby enabling congestion to set in. All passengers inherently opt for the shortest route in the network in order to save travelling time, which ultimately results in blocked roads and very slow traffic movements [6]. All these factors lead to under-utilization of a city's intricately-built road network.

Many countries adopt the "odd-even rule" [7] that allows only a section of the vehicles to operate on a given day, for e.g., vehicles with odd number plate on Monday. In the UK, they reduced congestion by creating "out of town" [8] commercial areas and pedestrianising city centres, thereby reducing driveable routes in the most populated areas. Many works involving autonomous systems, where a central operator can manage system-wide actions, have tried to increase the throughput of a network. [9, 10] have shown that autonomous vehicles may be able to drive faster and follow other vehicles at closer distances without compromising safety, thereby effectively increasing the capacity of a road. Others have suggested ride-sharing services such as UberPool and LyftLine, or public transport to lessen the traffic burden of the [11, 12, 13]. Rerouting empty vehicles to potential regions of higher customer requests based on historical data has also shown to lessen region-wise network congestion [14, 15].

Multi-class fleets [16] allow breaking a customer trip into 3 classes: first mile, middle mile and last mile. The customer can choose to either walk on pedestrian paths, use dedicated cycle-ways, or on-demand scooters for their first and last mile. The middle mile will cover at least 90% of the customer trips using the main road networks of a city and make use of fast-speed cars or public transports as per convenience. By using all three classes in an optimal combination, customers can maneuver through crowded paths and congested roads with ease. This also gives customers flexibility in choosing their preferred mode of transport based on accessibility and cost.

1.2 Problem Definition

In this project, we present an efficient algorithm to route customer trips in a congestion-aware manner using a multi-class fleet of vehicles. The trip is divided into multiple legs in order to decrease the overall travel-time of the customer. Each leg of the trip is served by a different mode of transport depending on travel time and ease of the customer. The algorithm provides optimal transit points for each trip, i.e., till what point should a person walk/cycle and then take a car/public transport, in order to get the least travel time for the entire trip. All customer requests are routed into system-optimal paths in order to get the least travel time for the entire system. The optimal points are however user-centric and can therefore, act as a recommendation or alternate path (see Fig. 1).

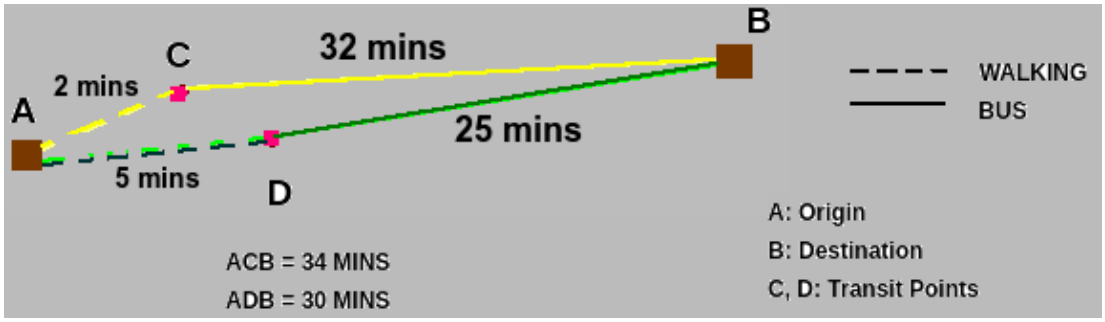

Figure 1: A multi-class combination of walking and bus, where D is the optimal transit point for minimum trip time.

2 Literature Review

1. [17] tackled the problem of congestion-aware routing for a network consisting of Autonomous Mobility on Demand (AMoD) systems as they allow for system-wide coordination. It showed that intelligently routing and rebalancing each vehicle on capacitated roads did not increase congestion for small demand sizes. However, customer trips were considered event-based and routed greedily using an A* approach. This results in a Wardrop User equilibrium [18] which is considered sub-optimal. Moreover, the solutions could not be used for a real-time analysis [19]. Therefore, our framework proposes an improvement by implementing an optimal Wardrop System equilibrium [20] with real congestion information of a city.

2. [21] proposes congestion-aware route-planning policies for AMoD systems under mixed traffic conditions. Customer and re-balancing flows are routed in a system-centric fashion. However, for high levels of demand, pure AMoD travel can be detrimental due to the additional traffic stemming from its rebalancing flows. Private vehicles are routed in a user-centric way in order to adapt to the AMoD flows, which in turn, hampers systems to reach an optimal solution. By combining AMoD rides with moderate levels of walking, the overall system performance improved by 50%. However, there are no optimal transit points present to guide when to stop walking and start using an AMoD system. We also provide route-recovery strategies for the flow solution.

3. In [16], multi-class fleets were introduced that serviced a request using a combination of heterogeneous vehicles (cars, buggies, scooter, walking). The optimal fleet size for individual class of vehicles were obtained using a genetic algorithm which was constrained on the overall budget of the multi-class AMoD system. Different vehicles were assigned to each customer sequentially and with the objective of minimizing the travel time across first, middle and last miles of individual customer journey. Due to this greedy approach, when the demand is much greater than the expected demand used to optimize the fleet size, the total average travel time for multi-class is worse than that of single-class. Hence the authors introduce, multi-class with walking mode where the customers are suggested to walk for the first and/or last miles of their journey to minimize their total travel time. Though the multi-class with walking mode performs better for unexpected demand sizes, it has fixed transit points between different modes of travel. We draw inspiration from this work and propose dynamic transit

point selection for congestion-aware routing.

4. Several research papers demonstrated new computationally efficient ways to replace the traditional process of the Traffic Assignment Problem (TAP). Frank Wolfe optimization [22] and Contraction Hierarchies [23] were used in [24] in order to achieve a system-optimal solution to the congestion-aware routing problem. There was a 20% improvement in computational time over the traditional TAP assignments. Even though the convergence rate of Frank Wolfe is known to be slow, its sub-steps could be easily made faster with parallel computing. In [25], the optimal meeting points are obtained using fast and efficient search for the shortest distance paths. The authors propose a hybrid algorithm that uses Euclidean distances as the shortest path heuristic. The time complexity for this is $O(n^2)$. It is highly useful in case of large combinatorial problems. We modify their hybrid technique to obtain near optimal and computationally efficient transit points between different modes of transport.

5. in [21], various cost functions such as the Bureau of Public roads (BPR), 2-line approximation, 3-line approximation, and Davidson's heuristics were compared. In case of convex problems, BPR was an effective choice. Since we are not accounting for rebalancing flows, our problem remains convex. We are using a dynamic BPR heuristic by integrating the traffic speed bands into its capacity term.

6. In [11], car-pooling with 2-3 riders per vehicle was introduced in order to serve more customers with lesser vehicles on the road. [12] presented a more general model for real-time high-capacity ride-sharing that had rider-capacity of up to 10 simultaneous customers per vehicle. Their results showed that 98% of the taxi rides currently served by over 13,000 taxis (of capacity one) could be served with just 3,000 taxis (of capacity four). This resulted in greater traffic throughput for increasing demand. However, trip assignments were still greedily user-centric causing the framework to be sub-optimal.

7. In [14], the network is optimally partitioned into re-balancing regions. Real-time demand estimate for every region is determined using incoming requests based on which idle vehicles are optimally assigned to these regions. There was a reduction in the average travel delay by 86%, the average waiting time by 37%, and the amount of ignored requests by 95% compared to [12] at the expense of an increased distance travelled by the fleet. The

algorithm could not incorporate the existing public transportation infrastructure and congestion information of the network.

8. In [26], ride-sharing was explored in terms of how many vehicles were needed, where they should be initialized, and how they should be routed to service all the demand. There was a reduction in the fleet size by 69% and travel delay of 1.7 mins for up to 2 customers per vehicle, and by 77% and 2.8 mins for up to 4 passengers per vehicle. However, it's computationally expensive for online adaptation.

9. [13] computed future demands of passenger requests, based on historical data. The predictions improved the positioning of the vehicles towards satisfying future requests, reduced waiting and travel time. This lessened region-wise congestion by performing re-balancing of the network. However, seasonal changes was not taken into account while predicting future requests and the analysis was static in nature.

3 Proposed Methodology

3.1 Timeline

The project started on February 1, 2020 by surveying all of the related work. We further moved onto setting up tools needed for the project including collection of data. We then analysed several path-planning approaches, developed novel algorithms, and validated them with experiments. The project ended on June 30, 2020 (see Fig. 2).

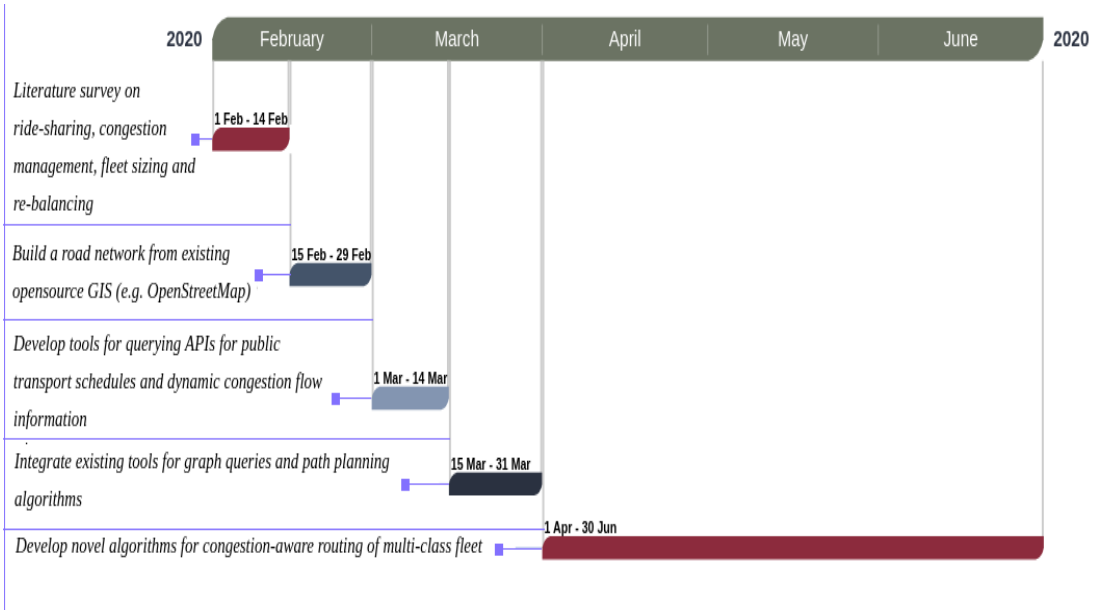

Figure 2: Timeline Gantt Chart

3.2 Method Overview

3.2.1 Phase 1: Setup

1. Setting up of Singapore city's road network (GIS) using the OpenStreetMap (OSM) database [27]. It consists of pedestrian paths, cycle-ways, and drive-able roads.

2. Collection of Traffic Speed Band dataset using LTA Datamall [28], an API for retrieving static and dynamic network information of Singapore.

3.2.2 Phase 2: Algorithm Design

1. Integrated the traffic speed band dataset with our OSM GIS and calculate congestion information for the entire network, time-wise.

2. Proposed an algorithm that finds optimal transit points for each mile (first, middle, last) for every customer request, such that the overall travel time for each customer is least.

3. Formulated a linear programming problem (LPP) that satisfies all origin-destination (OD) demands along with its capacity constraints, while minimizing travel time of the entire system.

3.2.3 Phase 3: Implementation

1. Created a set of OD trip demands and run the proposed algorithm to find optimal transit points from the first to the middle mile, and from the middle to the last mile.

2. Solved the LPP using Frank-Wolfe algorithm to get system-optimal flows for all demands for the middle mile.

3. Decomposed these flows into dedicated routes for each OD trip using an efficient shortest successive path algorithm [29].

An overview of the phases is presented in Fig. 3.

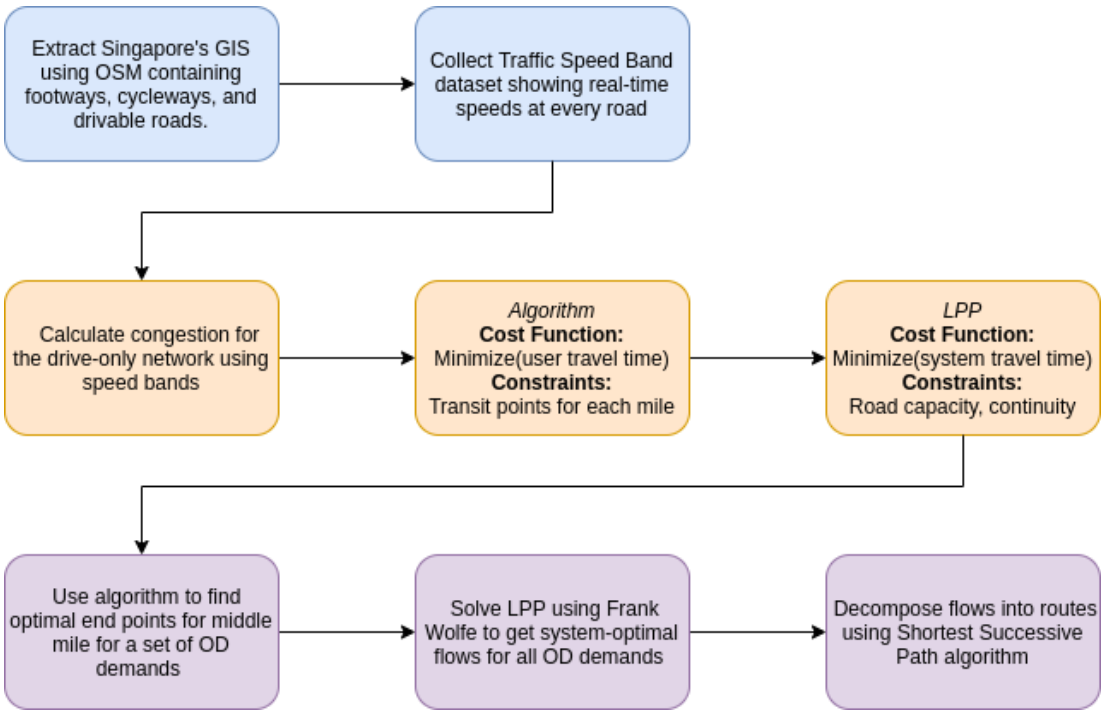

Figure 3: Project life cycle

4 Requirements

4.1 Data set

1. **Singapore's GIS:** Open Street Map (OSM) was used to extract Singapore's geographic information system (GIS) using the Overpass API.
   (a) The network is tagged with different types of roads or edges like pedestrian, cycle-ways, and drive-able routes. They constitute of corridors, cycle lanes, highways, service roads, etc.
   (b) The network is tagged with different types of nodes like bus stops, elevators, residential areas, parking lots, motorway junctions, etc.
   (c) Routing is possible from one node to another using a series of edges.
2. **Traffic Speed Band Data set:** It was collected from Land Transport Authority (LTA) Datamall owned by the Singapore Government.
   (a) It is a dynamic data set that updates itself every five minutes.
   (b) It shows speed bands (maximum and minimum observed speed) for every road link in Singapore giving real-time information about congestion on that link.
   (c) It was collected for two weeks including weekends, for these time slices: 09:00 A.M., 12:00 P.M., 03:00 P.M., 06:00 P.M., 09:00 P.M. .

4.2 Hardware and Software Requirements

1. A GPU with 12 GB RAM for fast computation .
2. **OSMnx 0.11.4:** Used to download, model, project, visualize, and analyze complex street networks from OpenStreetMap.
3. **NetworkX 2.4:** Used for path planning, routing and studying the structure and functions of Singapore's complex networks.
4. **Rtree 0.9.4:** Provided advanced spatial indexing features like Nearest neighbors search and Intersection search.
5. **Open Source Routing Machine(OSRM):** Its an open source router designed to compute the shortest path using contraction hierarchies or multi-level Dijkstra's.
6. **Python 3.6:** Interpreter language on top of which all these libraries would function.

5 Implementation

5.1 Collect and combine data

1. Extract Singapore's GIS using OpenStreetMap with the place name as *"Singapore, Central, Singapore"*. We get a drive-only network by specifying the network type as *"drive"*, and a pedestrian/cycle-way network by specifying the network type as *"walk"* or *"bike"*.The entire Singapore network can be visualized as a densely-connected directed graph $G(V, E)$ with V nodes (see Fig. 4) and E edges (see Fig. 5).

Figure 4: Nodes (V)

Figure 5: Edges (E)

2. Integrate the drive-only network with the traffic speed band data set using the *"Location"*, *"MaximumSpeed"*, and *"MinimumSpeed"* metadata (see Fig. 6). To incorporate temporal aspects, we use three slices of time: 12:00 A.M., 3:00 P.M., and 6:00 P.M.


```
[
    {
        "LinkID": "103000000",
        "Location": "1.3170142286123558 103.85298052044496 1.3166840028663076 103.85259882242372",
        "MaximumSpeed": "39",
        "MinimumSpeed": "30",
        "RoadCategory": "E",
        "RoadName": "KENT ROAD",
        "SpeedBand": 4
    },
]
```


Figure 6: Speed Band Dataset Snapshot

5.2 Compute congestion

Calculate congestion information for the entire drive-only network. It is calculated in terms of how much travel time is required to cross a road link in the network.

1. If vehicles present on a road link (u, v) are within the link capacity $c(u, v)$, they are assumed to be operating at free flow speed. Thus, the time taken for vehicles to cross a road link at free flow speed is called the free flow time $t(u, v)$. However, if vehicles present on a road link exceed the link capacity, congestion starts to set in. This lowers the the speed at which they are operating and hence, travel time to cross that link increases.

2. The travel time $t_d(u, v)$ along each edge is computed using a heuristic delay function that is related to the current volume of traffic on each edge. We use the Bureau of Public Roads (BPR) delay model [30], which computes the travel time on each edge (u, v) $\in$ edges (E) as:

$$t_d(u, v) := t(u, v) \left(1 + \alpha \left(\frac{f(u, v)}{c(u, v)}\right)^{\beta}\right)$$

where $f(u, v) := \sum_{m=1}^{M} f_m(u, v)$ is the total flow on edge (u, v), and α and β are usually set to 0.15 and 4 respectively.

5.3 Compute middle mile transit nodes

We now propose an efficient algorithm (see algorithm 1, 2) that will be used to find optimal transit points for each mile (first, middle, last) for every customer

request. Each customer request is composed of multiple legs(in this case, 3) served by different transportation modes (walking, cycle, taxi, public transport). The first and last miles can use only micro-mobility options like walking, cycling, or scooters (similar to [16]). Whereas only private taxis or public transport can be used for the middle mile (see Fig. 7).

Figure 7: A trip using a multi-class fleet

For example, a customer will walk (for 250m) to a certain node in the graph for the first mile, after which they would alight a bus (for 15 mins) until another node in the graph for the middle mile, and would bike further (for 400m) until their destination for the last mile. These transit points would be selected such that the overall travel time for each customer is the least. Thus, every customer trip would be like:

$$\mathbf{X} \longrightarrow \mathbf{A} \longrightarrow \mathbf{B} \longrightarrow \mathbf{Y}$$

X: Origin of customer request, start node of the first mile
A: End node of the first mile, start node of the middle mile
B: End node of the middle mile, start node of the last mile
Y: End node of the last mile, destination of customer request

X and **Y** are constant for every customer request. Therefore, we only need to compute **A** and **B**. In order to exercise maximum flexibility in choosing our combination of transit points while minimizing the travel times for every mile, we first route for the middle mile, i.e., ($\mathbf{A} \longrightarrow \mathbf{B}$) (see Fig. 8) such that the overall travel time for each customer is least.

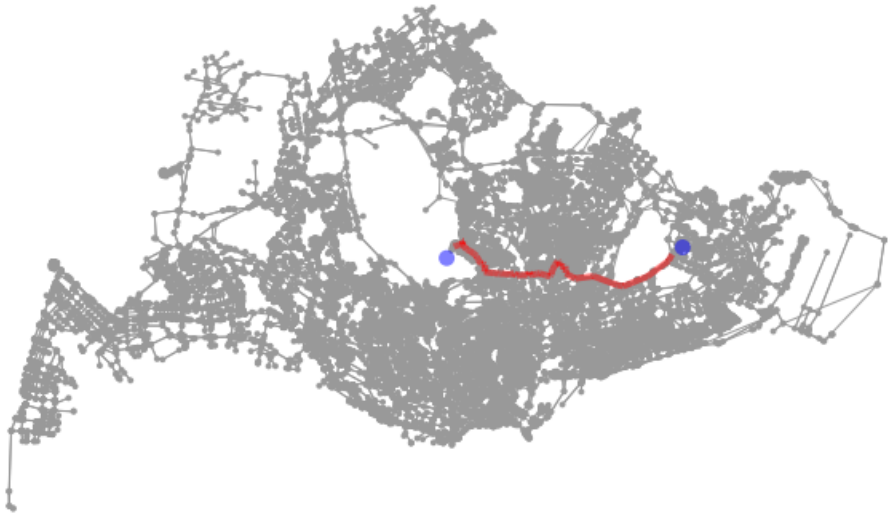

Figure 8: Routing shown for the middle mile (A —> B)

Algorithm 1: Find optimal transit nodes: A,B

Output: Demand set containing middle mile transit nodes

1: $Demand \leftarrow EmptyList()$;
2: $OD \leftarrow \{S, T\}$ {OD is set of all demands};
3: **for** $\{s, t\} \in OD$ **do**
4: $CandidateSources \leftarrow$ `GetNearestNodes(s, 720m)`;
5: $CandidateTargets \leftarrow$ `GetNearestNodes(t, 720m)`;
6: $EuclideanDict \leftarrow$ `SortLengthDict(`$CandidateSources, CandidateTargets$`)`;
7: $AB \leftarrow$ `ModifiedHybridSearch(`$EuclideanDict$`)`;
8: `Append` AB `to` $Demand$;
9: **end for**
10: $FirstMile \leftarrow$ `ShortestPath(`X, A`)`;
11: $LastMile \leftarrow$ `ShortestPath(`B, Y`)`;
12: **return** $Demand$

1. To find an optimal starting node **A** and ending node **B** for the middle mile, a heuristic based on the flexibility for the customer is made. According to it, all nodes within a 720 metre radius (euclidean distance) from the customer's starting and ending coordinates (**X**, **Y** respectively), are considered as potential sources and targets for the middle mile. This means that a customer is allowed to walk or cycle to/from their middle mile endpoints.

$$PotentialSources(X) = \{s_1, s_2, \ldots, s_n\}$$

$$PotentialTargets(Y) = \{t_1, t_2, \ldots, t_m\}$$

A total of $m*n$ combinations of {source, target} pairs are made. We choose that {source, target} pair as **A** and **B** which takes the least travelling time from *source* to *target* (see algorithm 1).

2. We use a modified version of the hybrid algorithm used in [25] to query through the set of {source, target} pairs. The algorithm uses a base distance with which it compares all other distances to find the shortest path, in order to save time on querying. We impose an extra cutoff on the shortest path search by using an extension of the Secretary problem [31], i.e., we stop the search for the shortest path after at most 37% of {source, target} pairs have been seen.

3. We use euclidean lengths as base distances for comparison with real lengths. Since our parameter is travel time, we calculate the euclidean length between each {source, target} pair and divide it by the permissible average speed of vehicles in Singapore (50 km/hr).

4. To calculate the real-time travel time of each {source, target} pair route, we use Contraction Hierarchies with edge weights/cost as the Bureau of Public Roads delay heuristic. This outputs a {source, target} pair with the least travel time between them, given real-time congestion information (see

algorithm 2).

Algorithm 2: Modified Hybrid Search

```
Data: Dictionary X with key:(A,B); value: Euclidean distance travel
      time, sorted in increasing order of value
Result: Nodes (A,B) s.t. travel time between them is least
Cutoff ← Floor(n/e); // Only search till first 37% of entries
(Source, Dest) ← (X[1][Source], X[1][Dest]);
minTime ← ∞;
for i ← 1 to Length(X) do
    travelTime ← ContractionHierarchies(Source, Dest);
    if i > Cutoff then
        if travelTime < minTime then
            Return current (Source, Dest) as its optimal;
        else
            Return (Source, Dest) optimal till now;
        end
    else
        if travelTime < minTime then
            Current (Source, Dest) is optimal till now;
            minTime ← travelTime;
            if i < Length(X) then
                if travelTime < X[i+1][Travel Time] then
                    Return current (Source, Dest) as its optimal;
                else
                    (Source, Dest) ← (X[i+1][Source], X[i+1][Dest]);
                end
            else
                Return current (Source, Dest) as its optimal;
            end
        else
            if i < Length(X) then
                if minTime < X[i+1][Travel Time] then
                    Return (Source, Dest) optimal till now;
                else
                    (Source, Dest) ← (X[i+1][Source], X[i+1][Dest]);
                end
            else
                Return (Source, Dest) optimal till now;
            end
        end
    end
end
```

5.4 Compute system-optimum flows for middle mile

In order to convert this framework into a social model and satisfy all OD demands, we formulate a linear programming problem. The objective or cost function is to minimize the travel time of the entire network, given a set of customer requests M such that, $m \in M$, where $m := (start, target)$. The capacity of the roads is used as a constraint for the upper bound on number of vehicles that can be at an edge E at the same time. Hence, the LP problem formulates as [17]:

$$\begin{aligned}
\text{minimize:} \quad & \sum_{m \in M} \sum_{(u,v) \in E} t(u,v) f_m(u,v) \\
\text{subject to:} \quad & \sum_{u \in V} f_m(u, s_m) + \lambda_m = \sum_{w \in V} f_m(s_m, w), && \forall m \in M \\
& \sum_{u \in V} f_m(u, t_m) = \lambda_m + \sum_{w \in V} f_m(t_m, w), && \forall m \in M \\
& \sum_{u \in V} f_m(u, v) = \sum_{w \in V} f_m(v, w), && \forall m \in M, v \in V \backslash \{s_m, t_m\} \\
& \sum_{m \in M} f_m(u, v) \leq c(u, v), && \forall (u, v) \in E
\end{aligned}$$

Constraints (2), (3) and (4) enforce continuity of each trip (i.e., flow conservation) across nodes. Finally, constraint (5) enforces the capacity constraint on each link.

Solving this LPP is NP-Hard. Hence we make use of a conditional gradient descent algorithm, Frank-Wolfe Optimization [32] using PyTrans. The OSM GIS is converted into *.tntp* [33] format. The solution of Frank-Wolfe optimization is feasible integral customer flows on congestion-free road links satisfying all demand. It is to note that these customer flows belong solely on the drivable network, since congestion is being monitored only for vehicles and public transport. These flows can be then decomposed into routes using a flow routing algorithm.

5.5 Decompose system flows into path

In order to decompose customer flows into dedicated routes:

1. The system-optimal flows act as the new capacity of the network. These capacities are used as an upper bound on the number of vehicles allowed on a road at once in the network.

2. All demand is added in a FIFO queue as per their time of arrival. Each customer is routed using a capacity-constrained weighted Dijkstra algorithm. [29] (see algorithm 3).

Algorithm 3: Capacity-bound weighted Dijkstra

```
Data: source, target
Result: System-optimal route for an OD trip
paths ← {source: [source], target: empty_list()};
Q ← Node Priority Queue;
for each vertex u in Graph do
    time[u] ← ∞; // Initialize lookup table storing travel time
       from source to u
end
time[source] ← 0; // Travel time from source to itself is 0
Q.add_with_priority(source, time[source]);
while Q is not empty do
    u ← Q.pop_min();
    for each neighbor v of u do
        if capacity(u, v) > 0 then
            time_uv ← BPR(u, v); // Heuristic travel time
            alt ← time[u] + time_uv;
            if alt < time[v] then
                time[v] ← alt;
                Q.add_with_priority(v, time[v]);
                paths[v] ← path[u].append(v)
    end
end
return paths[target]
```

5.6 Compute first and last miles

To compute the first and last mile routes, only pedestrian roads and cycle-ways are used for routing. The customer is expected to travel these routes by either walking, cycling, or using on-demand scooters (similar to [16]). These distances are calculated keeping in mind that a customer may want to (i) walk for a maximum of 10 minutes or 720 metres (assuming walking speed as 1.2 m/s), or (ii) cycle for a maximum of 15 minutes or 2 Km (assuming cycling speed as 5.4 m/s).

1. For the first mile, the starting node is **X**. The customer is expected to reach from $\mathbf{X} \longrightarrow \mathbf{A}$ as part of their first mile journey. This route is calculated using weighted-Dijkstra using edge weights as length of the paths.

2. For the last mile, the starting node is **B**. A route from **B** to **Y** is calculated using weighted-Dijkstra with path length based weighted edges. The

customer is expected to reach from $\mathbf{B} \longrightarrow \mathbf{Y}$ in order to finish their entire trip.

6 Results

All numerical experiments are done on 3 time slices keeping in mind the traffic patterns of Singapore: 12:00 A.M. (off-peak), 3:00 P.M. (moderately-peak), and 6:00 P.M. (high-peak). These experiments showcase the spatial as well as the temporal aspects of our proposed framework.

6.1 Minimum distance path V.S. Minimum travel-time path

A set of 1000 OD trips is taken. (i) The blue bars represent the time taken to travel on its shortest-distance path, and (ii) The orange bars represent the time taken to travel on the least congested path (see Fig. 9).

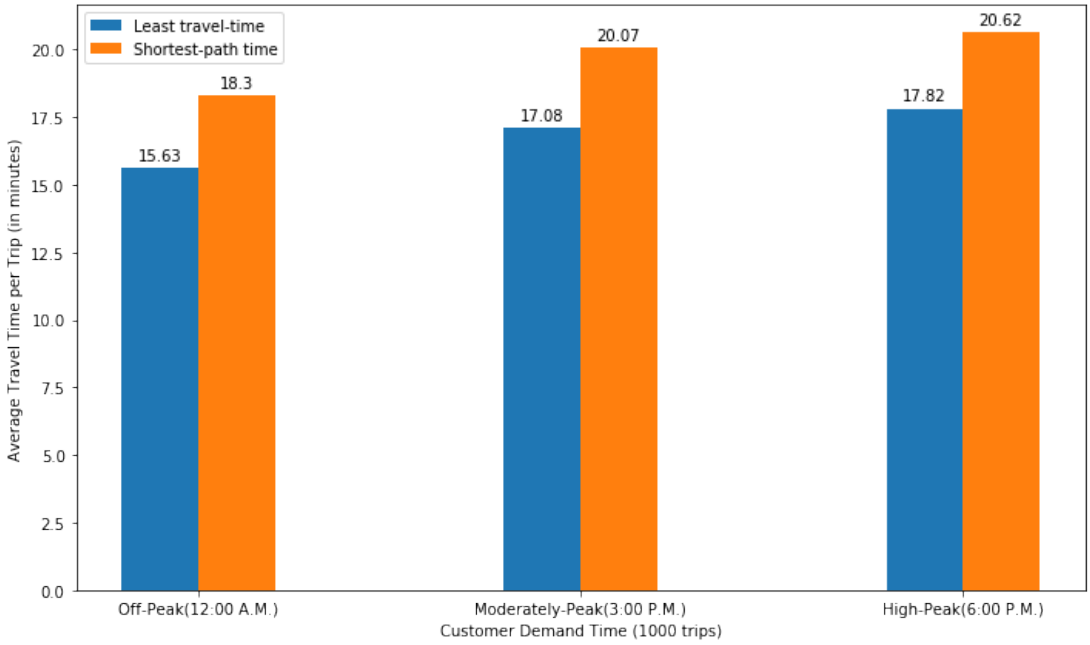

Figure 9: Shortest path length travel-time VS Optimal travel-time

The capacities of roads decrease with increasing peak times due to the increasing number of vehicles in the network. Hence, this results in increase in travel times for the same routes. Majority of customers inherently opt for the shortest-distance route because of lack of congestion information, thereby increasing their trip times drastically. The average increase in trip times by taking a shortest-distance path instead of the least congested path is 14.34%. Therefore, the shortest-distance paths don't always correspond to minimum travel-time paths.

$$\begin{aligned} &\textbf{\% Average travel-time increase} \\ &= \textbf{(17.08 + 17.51 + 15.71)/3} \\ &= \textbf{16.77\%} \end{aligned}$$

6.2 Modified Hybrid Search V.S. Hybrid search V.S. Exhaustive search

Given a set of OD pairs of size N, such that,

$$OD = \{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$$

we seek to find an origin-destination pair (s, t) such that the travel-time between them is minimum. A set of 100 customer trips is taken and 3 different search methodologies are compared in terms of accuracy and performance.

(i) Exhaustive Search looks at the travel times of every $(s_i, t_i) \in OD$. (ii) Hybrid Search uses euclidean distances as a base comparison for different (s_i, t_i). (iii) Modified Hybrid Search extends on the concept of Hybrid Search but adds a depth cutoff on its search strategy. This cutoff is taken to be as (N/e), where N is the depth of the list to be searched and e is base of natural logarithm [31], thereby always having an optimal win probability of at least 37%.

1. **Travel-time deviations:**

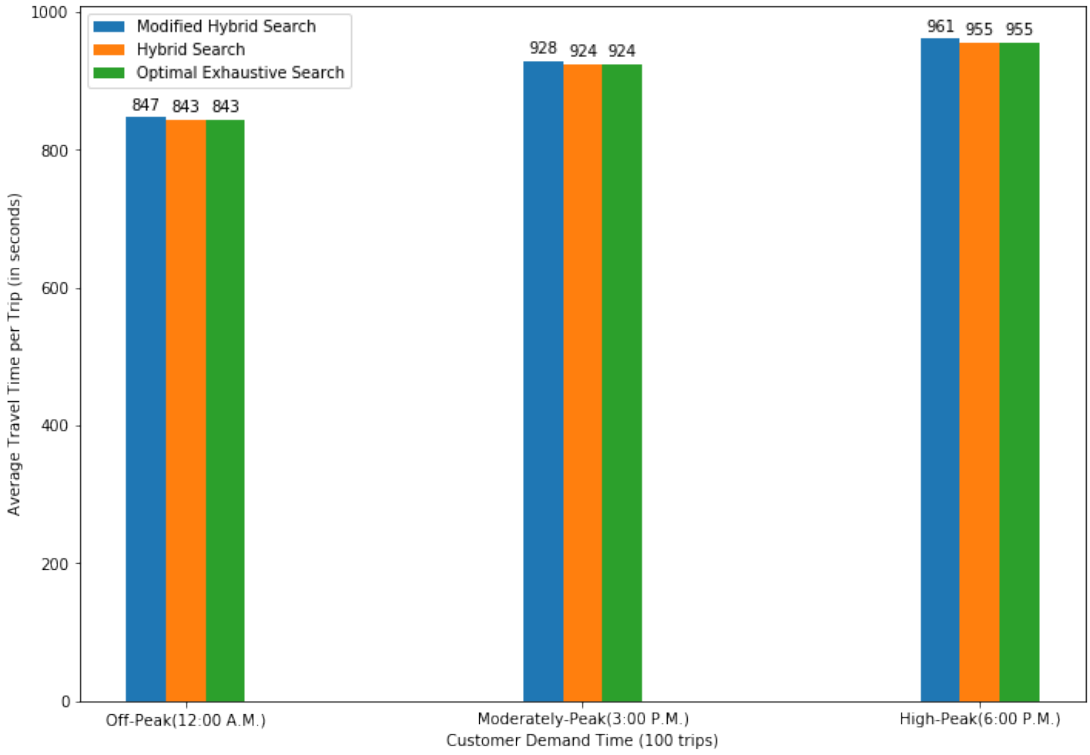

Figure 10: Travel-times: Modified Hybrid Search VS Hybrid Search VS Exhaustive Search

Figure 10 shows that Exhaustive Search and Hybrid Search both output an OD pair having minimum travel time. Hence, both of these searches are optimal. However, Modified Hybrid Search doesn't always output an

optimal OD pair. The average deviation in travel times between the 3 searches is 4.67 seconds. Since, this deviation is very less with respect to traffic times, Modified Hybrid Search can be considered as near-optimal.

$$\textbf{Average travel-time deviation (in secs)} = \textbf{(4 + 4 + 6)/3} = \textbf{4.67 seconds}$$

2. **Number of queries:**

   Since exhaustive search strategies are optimal but computation heavy, we seek to see if Modified Hybrid Search can provide faster computation for near-optimal results. We tabulate the number of queries made to the database server by each of the 3 search strategies (see Fig. 11). Finding distance between two nodes constitute a query.

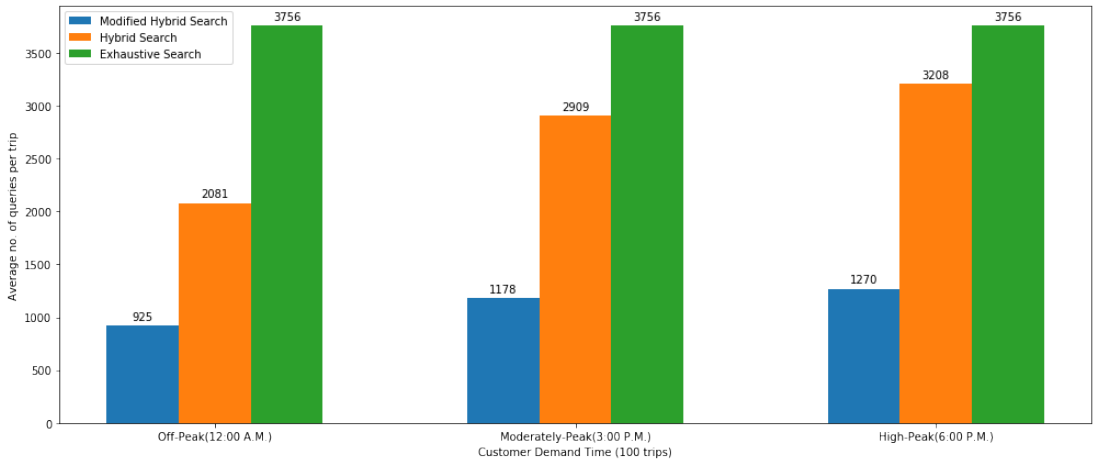

Figure 11: No. of queries: Modified Hybrid Search VS Hybrid Search VS Exhaustive Search

   (a) Exhaustive search iterates through every $(s_i, t_i) \in OD$, thereby finding a solution in $O(N)$.

   (b) Hybrid search stops its iterations if the current (s_i, t_i) pair's distance is lesser than its next base distance, or if it has reached the end of the set. Hence, sometimes it takes $< N$ iterations to find a solution. Thus, its worst-case time complexity is also $O(N)$.

$$\textbf{\% queries saved (Hybrid, Exhaustive)} = \textbf{(44.6 + 22.55 + 14.59)/3} = \textbf{22.25\%}$$

(c) Modified Hybrid search stops its iterations if the current (s_i, t_i) pair's distance is lesser than its next base distance, or if it has reached the depth cutoff. Hence, sometimes it takes $< (N/e)$ iterations to find a solution. Thus, its worst-case time complexity is *O(N/e)*.

$$\begin{aligned}&\textbf{\% queries saved (Modified Hybrid, Exhaustive)}\\ &= \textbf{(75.37 + 68.64 + 66.19)/3}\\ &= \textbf{70.07\%}\end{aligned}$$

Hence, it shows that Modified Hybrid Search outperforms the traditional Exhaustive Search and Hybrid Search, and is highly efficient with negligible average delay for this use case.

6.3 Multi-Class System Equilibrium V.S. Single-Class User Equilibrium

A set of 5 batches consisting of 500 OD trips each is taken. All demands originate and end in Downtown Singapore, which is the busiest region of the city for majority of the day. All of these are routed either using a (i) single class of vehicle (e.g. point-to-point routing) with a user-centric approach, (ii) or a combination of biking and taxi (multi-class) with a system-centric approach.

After decomposing the system optimal flows into routes using capacity-constrained Dijkstra (Algorithm 3), we record different cost parameters and how many of these demands are infeasible. Since, all the decomposed routes follow system-optimal flow constraints, they are bound to be within capacity. This means they all travel in a congestion-free network.

1. **Flow-Time Cost:**

   We address the effect of incorporating micro-mobility options in one's journey for reasonable distances (maximum 1 KM) with respect to the overall system cost. Since no vehicles are needed during the first and last mile in a multi-class framework, flow-time cost is only calculated for the middle mile of a customer trip.

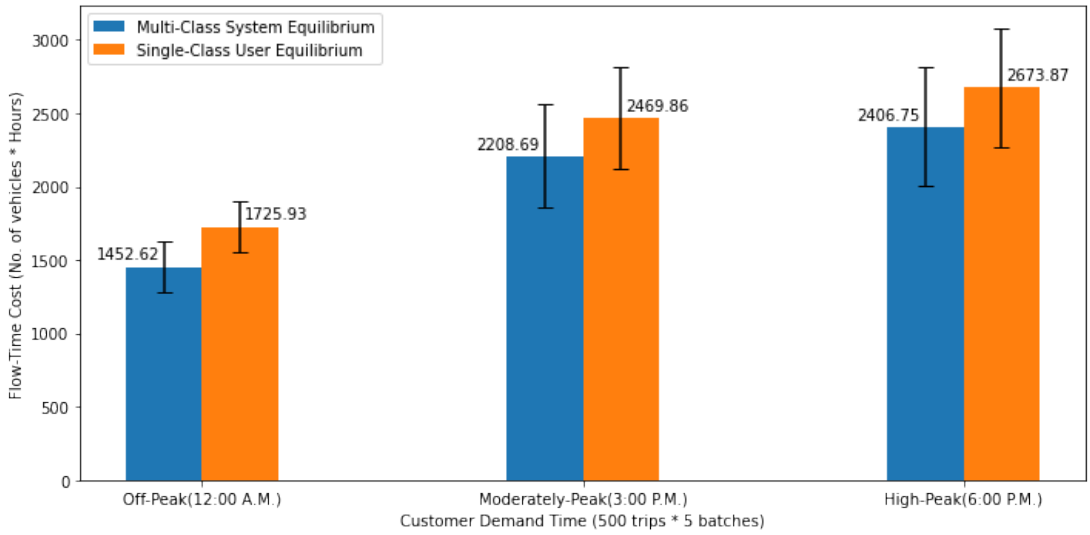

Figure 12: Average Flow-Time Cost

Figure 12 shows that the average reduction in the overall system cost by using a multi-class fleet of vehicles over just a single vehicle is 12.13%. This translates to 33 minutes per trip request. This shows a drastic potential decrease in congestion in the entire network by pedestrianizing a very small % of the entire trip (in this case up to 1KM).

$$\begin{aligned} &\textbf{\% Average Flow-Cost Reduction} \\ &= \textbf{(15.84 + 10.57 + 9.99)/3} \\ &= \textbf{12.13\%} \\ &\tilde{=} \textbf{ 33 minutes/trip} \end{aligned}$$

2. **Waiting Queue Size:**

   We seek to evaluate the congestion by assigning routes to all 500 customers using different flow equilibrium(s). After decomposing the flows into routes for every demand, network capacities were reached early, thereby leaving some user demands as infeasible. Hence, these trips are added to a waiting queue and would be served in the next batch-process.

   Since all routes follow capacity bounds, all feasible demands correspond to congestion-free demands. Hence, there is 0% congestion at all times in the network. Figure 13 shows that there are much greater infeasible trips for user equilibrium flows than system equilibrium. This means that network capacities are reached faster in user equilibrium, resulting in parts of road network being blocked. Hence, this results in the onset of congestion and a larger fraction of user demands is not served.

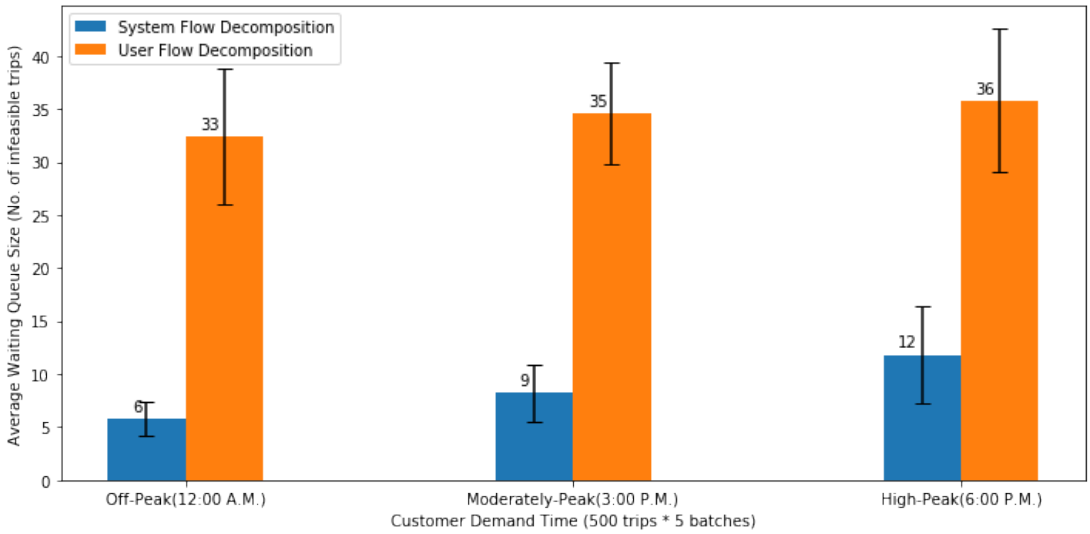

Figure 13: Average Waiting Queue Size

$$\begin{aligned}&\textbf{\% Average Waiting Queue Size Reduction}\\&= \textbf{(81.81 + 74.29 + 66.67)/3}\\&= \textbf{74.26\%}\end{aligned}$$

3. **Travel-Time Cost:**

   All demands are fed into the Frank-Wolfe algorithm to find flows that correspond to either (i) system equilibrium or (ii) user equilibrium. Current frameworks opt for user equilibrium flows as it tends to compute shortest routes greedily. However, this does not adhere to a social framework as it does not take into account the flows originating from other customer demands. Whereas, a system equilibrium keeps in mind the current congestion information, and future possible congestion from current customer demands. We seek to see the effect of these equilibrium(s) on the overall travel-time cost of the network.

   Figure 14 shows slightly higher travel times for our framework. This is because of the usage of bicycles/walking in first and last mile of the trip, which takes longer to cover than by simply using a motor vehicle. However, the increase in travel-time is extremely less at the cost of serving a higher number of trips.

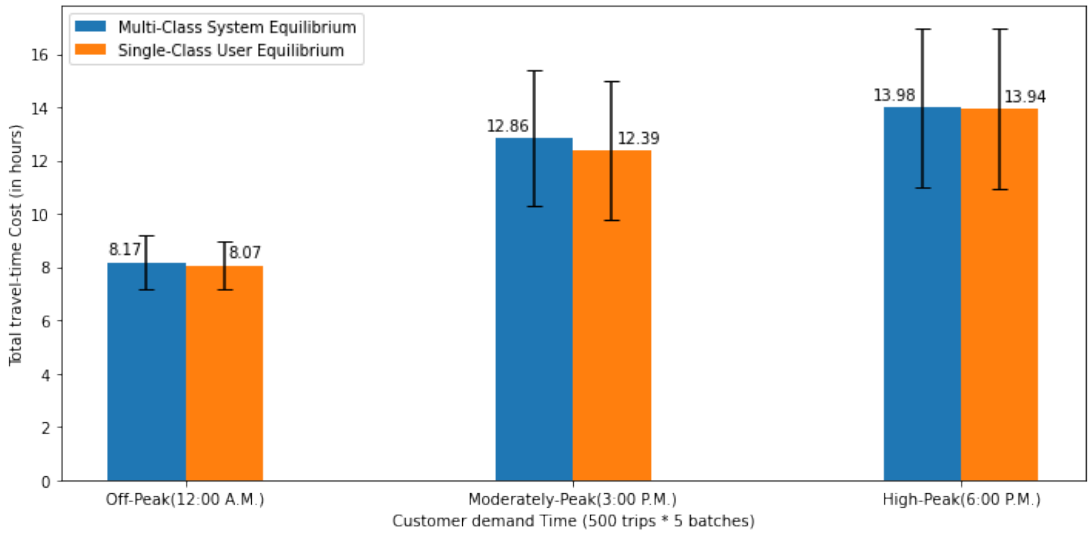

Figure 14: Average Travel-Time cost

$$\textbf{Average Travel-time Increase per 500 trips (in minutes)}$$
$$= \textbf{(6 + 28.2 + 2.4)/3}$$
$$= \textbf{12.2 minutes}$$

Figures 12, 13, and 14 show that a multi-class system optimal framework is cost-effective, results in greater network utilization and is a social model for traffic assignment compared to the current single-class user-centred framework.

7 Conclusion and Future Scope

In this project, we studied the achievable benefits of using a centrally-controlled multi-class setup for providing congestion-aware routes to customers. We proposed a fast and efficient Modified Hybrid Search algorithm to find transit nodes between this inter-modal service that accounts for reasonable walking/cycling distances. Our algorithm reduces the number of database queries by 70.07%, minimizes the overall flow-time cost of the system, and provides shortest possible routes in a capacity-bound network. Our experimental results support our hypothesis that with a small increase in the expected travel time, we can maximize the network-serving efficiency by 74.26%. In conclusion, we successfully built a social framework that performs congestion-aware routing for multi-class mobility-on-demand services and proves better than state-of-the-art.

This work opens the field for several extensions in future. On the computational side, we mention that the framework's performance can be further improved by parallel computing of the *All or Nothing* subroutine in Frank-Wolfe optimization. Moreover, a significant speedup can be gained from using a multi-core machine while executing the Modified Hybrid Search algorithm, as all transit points are chosen in a user-optimal way.

On the implementation side, we seek to see the performance of this framework after adapting to a fully AMoD setting, or one with mixed traffic conditions [21]. We also plan to investigate approaches that account for rebalancing flows or offer any ride-sharing capabilities for the middle-mile. Finally, our goal is to predict expected travel-times using historic speed band datasets by incorporating machine learning techniques such as vectorized feature selection, thereby enabling the model to be used in real-time.

References

[1] R. W. Caves, "Encyclopedia of the city," *Routledge p.141*, 2004.

[2] B. Templeton, "Traffic congestion and capacity," *http://www.templetons.com/brad/robocars/congestion.html*, 2010.

[3] W. Mitchell, B. Borroni, and L. Burns, "Reinventing the automobile: Personal urban mobility for the 21st century," *MIT Press*, 2010.

[4] K. Spieser, K. Treleaven, R. Zhang, E. Frazzoli, D. Morton, and M. Pavone, "Toward a systematic approach to the design and evaluation of autonomous mobility-on-demand systems: A case study in singapore. in: Road vehicle automation," *Springer International Publishing, Cham, Switzerland*, 2014.

[5] M. Pavone, S. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *Int Journal of Robotics Research*, 2012.

[6] M. Barnard, "Autonomous cars likely to increase congestion," *http://cleantechnica.com/2016/01/17/autonomous-cars-likely-increase-congestion*, 2016.

[7] H. Ergo, "Implementation of odd-even rule in new delhi," *https://www.hdfcergo.com/blogs/car-insurance/implementation-of-odd-even-rule-in-new-delhi*, 2019.

[8] I. UK, "A car-free future? how uk cities are moving towards a pedestrian age," *https://www.independent.co.uk/environment/car-free-cities-pedestrianisation-cycling-driverless-vehicles-york-oslo-birmingham-a9299856.html*, 2020.

[9] J. Pérez, F. Seco, V. Milanés, A. Jiménez, J. Diaz, and T. De Pedro, "An rfid-based intelligent vehicle speed controller using active traffic signals.," *Sensors 10(6):5872–5887*, 2010.

[10] R. Zhang and M. Pavone, "Control of robotic mobility-on-demand systems: a queuing-theoretical perspective," *Proceedings of Robotics: Science and Systems Con- ference, Berkeley, CA*, 2014.

[11] P. Santi, "Quantifying the benefits of vehicle pooling with shareable networks," *Proc Natl Acad Sci USA 111(37):13290–13294*, 2014.

[12] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences of the United States of America*, 2017.

[13] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive routing for autonomous mobility-on-demand systems with ride-sharing," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[14] A. Wallar, J. Alonso-Mora, D. Rus, and M. van der Zee, "Vehicle rebalancing for mobility-on-demand systems with ride-sharing," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[15] K. Spieser, S. Samaranayake, W. Gruel, and E. Frazzoli, "Shared-vehicle mobility-on-demand systems: A fleet operator's guide to re-balancing empty vehicles," *Transportation Research Board 95th Annual Meeting, January Washington, DC*, 2016.

[16] M. Meghjani, S. D. Pendleton, K. Anna, Marczuk, Y. Hong Eng, X. Shen, M. H, J. Ang, and D. Rus, "Multi-class fleet sizing and mobility on demand service," *Comples Systems Design Management Asia*, 2018.

[17] F. Rossi, R. Zhang, Y. Hindy, and M. Pavone, "Routing autonomous vehicles in congested transportation networks: structural properties and coordination algorithms," *Springer Link: Autonomous Robots (May)*, 2018.

[18] J. G. Wardrop, "Some theoretical aspects of road traffic research," *Proceedings of the Institution of Civil Engineers*, 1952.

[19] V. Pillac, M. Gendreau, C. Guéret, and A. Medaglia, "A review of dynamic vehicle routing problems," *Eur J Oper Res*, 2013.

[20] Wikipedia, "Wardrop equilibria," *https://en.wikipedia.org/wiki/John-Glen-Wardrop*, 1952.

[21] S. Wollenstein-Betech, A. Houshmand, M. Salazar, M. Pavone, C. Cassandras, and I. Paschalidis, "Congestion-aware routing and rebalancing of autonomous mobility-on-demand systems in mixed traffic," *https://arxiv.org/abs/2003.04335*, 2020.

[22] M. Frank and P. Wolfe, "Frank-wolfe algorithm," *Frank-Wolfe algorithm Wikipedia*, 1956.

[23] J. Dibbelt, B. Strasser, and D. Wagner, "Customizable contraction hierarchies," *Journal of Experimental Algorithmics*, 2016.

[24] K. Solovey, M. Salazar, and M. Pavone, "Scalable and congestion-aware routing for autonomous mobility-on-demand via frank-wolfe optimization," *https://arxiv.org/abs/1903.03697*, 2019.

[25] M. Meghjani, S. Manjanna, and G. Dudek, "Fast and efficient rendezvous in street networks," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[26] A. Wallar, J. Alonso-Mora, and D. Rus, "Optimizing vehicle distributions and fleet sizes for mobility-on-demand," *International Conference on Robotics and Automation (ICRA)*, 2019.

[27] OpenStreetmap and Wiki, "Openstreetmaps," *www.openstreetmap.org/*, 2004.

[28] Government, of, and Singapore, "Lta datamall," *mytransport.sg/content/mytransport/home/dataMall/*, 2020.

[29] TopCoder, "Data science algorithms: Successive shortest path," *topcoder.com/community/competitive-programming/tutorials/minimum-cost-flow-part-two-algorithms/*, 2016.

[30] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems, Science, and Cybernetics*, 1968.

[31] P. Freeman, "The secretary problem and its extensions: A review," *International Statistical Review / Revue Internationale de Statistique*, 1983.

[32] Daisik, "Pytrans/urban network analysis," *https://github.com/PyTrans/Urban-Network-Analysis*, 2018.

[33] T. N. for Research Core Team, "Transportation networks for research," *https://github.com/bstabler/TransportationNetworks*, 2016.