On-demand High-capacity Ride-sharing via Dynamic Trip-Vehicle Assignment with Rebalancing

by

Alexander James Wallar

BSc (Hons), University of St Andrews (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author . Signature redacted .....
Department of Electrical Engineering and Computer Science
May 19, 2017

Certified by .. Signature redacted .....
Daniela L. Rus
Professor
Thesis Supervisor

Accepted by .. Signature redacted .....
Leslie A. Kolodziejski
Chair, EECS Committee on Graduate Students

On-demand High-capacity Ride-sharing via Dynamic Trip-Vehicle Assignment with Rebalancing

by

Alexander James Wallar

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2017, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

On-demand ride-sharing systems with autonomous vehicles have the potential to enhance the efficiency and reliability of urban mobility. However, existing ride-sharing algorithms are unable to accommodate high capacity vehicles and do not incorporate future predicted demand. This thesis presents a real-time method for high-capacity ride-sharing that scales to a large number of passengers and trips, dynamically generates optimal routes with respect to online demand and vehicle locations, and incorporates predictions of anticipated requests to improve the performance of a network of taxis. We experimentally assess the trade off between fleet size, capacity, waiting time, travel delay, and amount of predictions for low to medium capacity vehicles. We validated the algorithm with over three million taxi rides from the New York City taxi dataset and demonstrate that our approach can service nearly 99% of Manhattan taxi demand using a fleet of only 3000 vehicles (less than 25% of the active taxis in Manhattan).

Thesis Supervisor: Daniela L. Rus
Title: Professor

Acknowledgments

I would like to thank my advisor, Daniela Rus, for all her support and guidance and Javier Alonso-Mora for bringing me onto this research.

I would also like to thank my family for their unconditional love and encouragement. I can never repay them for the opportunity they have provided me to pursue higher education.

Lastly, I'd like to thank my wife Jessica. If it wasn't for Jess, I would never be at MIT to begin with. I dedicate this thesis to Jess.

Contents

List of Figures

Chapter 1

Introduction

New user-centric services are transforming urban mobility by providing timely and convenient transportation to anybody, anywhere, anytime. These services have the potential for a tremendous positive impact on personal mobility, pollution, congestion, energy consumption and thereby quality of life. The cost of congestion in the United States alone is roughly $121 billion per year or one percent of GDP [38], which includes 5.5 billion hours of time lost to sitting in traffic and an extra 2.9 billion gallons of fuel burned. These estimates do not even consider the cost of other potential negative externalities such as the vehicular emissions (greenhouse gas emissions and particulate matter) [33], travel-time uncertainty [11] and a higher propensity for accidents [24]. Recently, the large-scale adoption of smart phones and the decrease in cellular communication costs has led to the emergence of a new mode of urban mobility, namely mobility-on-demand (MoD) systems, led by companies such as Uber, Lyft and VIA. These systems are able to provide users with a reliable mode of transportation that is catered to the individual and improves access to mobility to those who are unable to operate a personal vehicle, reducing the waiting times and stress associated with travel.

One of the major inefficiencies of current MoD systems is their capacity limitation, typically restricted to two passengers or the ride is not able to be shared to begin with. Our method applies not only to shared taxis, but also to shared vans and mini-buses. This is a difficult extension to current work due to the computational

costs associated with solving such large optimization problems. A recent study in New York City showed that up to 80 percent of the taxi trips in Manhattan could be shared by two riders with an increase in the travel time of a few minutes [37]. However, this analysis was i) limited to two riders for an optimal allocation (three with heuristics), ii) intractable for larger number of passengers, and iii) did not allow for allocation of additional riders after the start of a trip. There are no studies of this scale that quantify the benefits of larger-scale ride pooling, mainly due to the lack of efficient and scalable algorithms for this problem, both of which we address in this work.

Here we consider the problem of using a fleet of vehicles with varying passenger capacities, addressing both the problems of assigning vehicles to matched passengers and rebalancing -or repositioning- the fleet to service demand. In contrast, [37] ignored the vehicle assignment and rebalancing problems and was unable to assign passengers to ongoing trips. We show how the unified problem of passenger and vehicle assignment can be solved in a computationally efficient manner at a large-scale, thereby demonstrating the capability to operate a real-time MoD system with multiple service tiers (shared-taxi, shared-vans and shared-buses) of varying capacity.

Efficient algorithms capable of assigning travel requests to a fleet of vehicles, and routing the vehicles efficiently, are required. In this work, we present a constrained optimization method which accounts for future, predicted, requests to route the vehicles. Based on historical data, we first describe a method to compute a probability distribution over future demand. Then, we describe an any-time optimal method for vehicle routing and passenger assignment that takes into account the future demand to produce routes and assignments that in expectation reduce the travel and waiting times. Our method can assign thousands of requests to thousands of autonomous vehicles in real time, where we allow that several passengers with independent trips share a vehicle and that a vehicle picks additional passengers as it progresses in its route.

We quantify experimentally the performance trade-offs between fleet size, capacity, waiting time, travel delay, and operational costs for low and medium capacity vehicles

(such as taxis, vans or mini-buses) in a large urban setting. Detailed experimental results are presented for a subset of approximately 3 million rides extracted from the New York City taxicab public dataset. We show that 3000 vehicles of capacity 2 and 4 could serve 94% and 98% of the demand with a mean waiting time of 3.2 and 2.7 minutes, and a mean delay of 1.5 min and 2.3 min, respectively. To achieve 98% service rate, with comparable waiting time (2.8 min) and delay (3.5 min) a fleet of just 2000 vehicles of capacity 10 was required. This is 15% of the active taxis in NYC. We also show that our approach is robust with respect to the density of requests and could therefore be applied to other cities. A video is available at `https://youtu.be/EzOWFu7fMD0`.

We also illustrate experimentally the benefit of reactive vehicle rebalancing and predictive positioning. We compare the waiting time, service rate, travel delay, distance travelled, percentage of shared rides, and computational time when using predictive positioning based on historical taxi demand. We show that we are able to enhance the rider experience by reducing the waiting time and travel delay by incorporating predicted future requests into our assignment algorithm.

Our system runs in real-time and is particularly suited to autonomous vehicle fleets that can continuously reroute based on real-time requests. It can also rebalance idle vehicles to areas with high demand and is general enough to be applied to other multi-vehicle, multi-task assignment problems. In practice our request-vehicle assignment algorithm would be able to drastically reduce the number of vehicles needed to service current taxi demand. In doing so it would reduce taxi related congestion, cut harmful emissions, and improve the commute time of passengers in busy urban centers. This work was published in the Proceedings of the National Academy of Sciences [5].

1.1 Contribution

In this thesis, we present an efficient constrained optimization method for vehicle routing and multi-request multi-vehicle assignment that takes into account anticipated future demand. We describe a method to predict future requests based on

historical taxi trip data. The main contribution is an anytime optimal algorithm that takes into account the predicted future demand to influence both the vehicle routing and the assignment of request to vehicles. The method works in the context of ride sharing, and extends the planning horizon beyond the current requests. We also provide extensive experimental results using over three million real trip requests from the New York City taxi dataset [18] to show the effectiveness of the algorithm. Below is a detailed list of contributions.

- A method for predicting taxi demand using a large data set of historical taxi trips
- An algorithm to optimally assign taxi requests to vehicles that allows for high capacity ride-sharing
- A system for using large scale historical data for measuring the effectiveness of ride-sharing systems
- Experimental analysis showing the trade-offs between fleet size, vehicle capacity, wait-time, travel delay, and servicing rate

This thesis combines material from two publications, [5] and [6], where [5] introduces the base algorithm for passenger assignment and vehicle routing and [6] introduces the for predicting future demand from historical taxi trip datasets and shows how these predictions can be incorporated into the assignment algorithm to improve its performance.

1.2 Organization

This thesis is organized as follows. Chapter 2 provides an overview of the related work. Chapter 3 describes how anticipated demand is predicted. Chapter 4 describes our algorithm for passenger assignment and vehicle routing. Finally, Chapter 5 assesses the performance of the algorithm with and without predictive positioning.

Chapter 2

Related Works

In this chapter, we discuss the work related to ride-sharing, mobility on demand, and taxi demand prediction. Sec. 2.1 gives a background on methods for predicting taxi demand, Sec. 2.2 describes prior work relating to vehicle routing problems and how it applies to ride-sharing, and Sec. 2.3 discusses current methods for ride-sharing and mobility on demand.

2.1 Predicting taxi demand

Informed driving is becoming a key feature to increase the sustainability of taxi companies and with a combination of readily available large datasets and powerful data mining tools, estimation of future patterns from data is an active field of research. Castro el al. used large scale GPS traces to construct traffic models used for prediction. Moreira-Matias et al. used a combination of time series forecasting techniques to predict the spatial distribution of taxi-passengers for a short time horizon using streaming data [32]. Gonzales et al. used GPS data from taxies to develop a model to determine the number of pickups and dropoffs for each hour of the day [23]. Chang, Tai, and Hsu developed a context-awrare demand hotspot prediction system that used data mining techniques to predict demand distributions with respect to time, weather, and taxi location [12]. Veloso et at. utilized a naive Bayesian classifier to determine the predictability of spatiotemporal distribution of taxi trips [43]. Thanks

to the large amount of data available on taxi trips in New York (over 100M a year), we adopt a frequentist approach to predict future demand and focus on its integration in the assignment and routing problem.

2.2 Vehicle routing

The transportation community has been studying vehicle routing problems that require servicing many concurrent demands subject to capacity and other constraints for many decades. For example, the Dynamic Traffic Assignment (DTA) problem [29, 46], where the system aims to maximize throughput by dynamically routing time-varying travel demands across the network. These types of problems can be solved using techniques such as mathematical programming [45], optimal control [20, 36], and simulation based methods [28]. The DTA problem assumes that the demand is serviced by an individual personal vehicle and rides are not shared. The problem of ride-sharing is more similar to the Vehicle Routing Problem with Pickup and Delivery (VRPPD) [17, 4, 31, 8, 22] where optimal routes need to be found to move passengers between certain pickup and delivery locations; the Vehicle Routing Problem with Time Windows (VRPTW) [26, 16, 10] where passengers need to be dropped off within time window constraints; the Capacitated Vehicle Routing Problem (CVRP) [7, 21] where vehicles have a limited passenger capacity; and the Dynamic Pickup and Delivery Problem [30, 9] where the demand is spatio-temporally distributed and must be serviced within specified time windows. However these problems do not consider how to optimally distribute the demand across the vehicles or how to pro-actively position the vehicles to anticipate future demand.

2.3 Ride-sharing and mobility on demand

Much of the fleet management literature for mobility-on-demand systems considers the case of ride-sharing without pooling requests, focusing on fluid approximations [34], queuing based formulations [44], case studies in specific regions (e.g., Sin-

gapore [40]) and operational considerations for fleet managers [39]. With the growing interest and rapid developments in autonomous vehicles, there is also an increasing focus on autonomous MoD systems [34, 14, 39]. However, none of these works considered the ride-pooling problem of servicing multiple rides with a single trip. The ride pooling problem is more related to the Vehicle Routing Problem and the Dynamic Pickup and Delivery Problem [42, 35, 9, 22, 41], where spatio-temporally distributed demand must be picked up and delivered within pre-specified time windows. A major challenge when addressing this problem is the need to explore a very large decision space, while computing solutions fast enough to provide users with the experience of real-time booking and service.

Previous approaches to this problem have focused on heuristic-based solutions [3, 25, 27, 2, 25, 19] to address the computational challenge that lye in assigning the potentially large fleet of vehicles to indiviual matched trips in an efficient manner. We present a reactive anytime optimal algorithm. That is, an algorithm that efficiently returns a valid assignment of travel requests to vehicles and then refines it over time converging to an optimal solution. If enough computational resources are available, the optimal assignment for the current requests and time would be found, otherwise, the best solution found so far is returned.

Traditional approaches that rely on an Integer Linear Program (ILP) formulation, such as [13], also provide anytime guarantees for the multi-vehicle routing problem. However, in contrast to our approach, their applicability is limited to small problem instances, which in [13] was 32 requests and four vehicles with a computation cost of several minutes. We also rely on an ILP formulation, but because we do not explicitly model the edges of the road network in the ILP, our approach scales to much larger problem instances. We observe that instances such as NYC, with thousands of vehicles, requests and road segments, can be solved in real time.

Our approach decouples the problem by first computing feasible trips from a pairwise shareability graph [37] and then assigning trips to vehicles. We show that this assignment can be posed as an Integer Linear Program (ILP) of reduced dimensionality. The framework allows for flexibility in terms of prescribing constraints such as

(but not limited to) maximum user waiting times and maximum additional delays due to sharing a ride. We also extend the method to pro-actively rebalance the vehicle fleet by moving idle vehicles to areas of high demand. In summary, we present a framework for solving the real-time ride pooling problem with (1) arbitrary numbers of passengers and trips, (2) anytime optimal rider allocation and routing dependent on the fleet location, and (3) online rerouting and assignment of riders to existing trips.

Chapter 3

Prediction of future demand

This chapter outlines our method for estimating anticipated demand. First, we discretize Manhattan into regions, then using the Manhattan taxi dataset, we employ a frequentist approach to estimate a probability distribution each region pair. Future requests are sampled from this probability distribution and incorporated into the batch assignment described in Chapter 4.

The chapter is structured as follows. Sec. 3.1 describes how we estimate the historical demand from the Manhattan taxi dataset. Sec. 3.1.1 outlines the process for partitioning Manhattan into discrete regions. Sec. 3.1.2 describes how a probability distribution for the pairwise demand is estimated using a frequentist approach and Sec. 3.1.3 describes how we sample from this probability distribution.

3.1 Estimation of historical demand

In a preprocessing step, the probability distribution of origin-destination requests is computed for fixed intervals of the week. We do so by discretizing the area into regions and cumulating requests from a year of historical taxi data.

Using a list of all intersections, we discretize the area into regions given by a distance parameter r which relates to the acceptable distance a person would need to walk. With this discretization, we can then assign the origin and destination of the requests from the historical data to the closest region centers. Using a frequentist

Figure 3-1: Region centers determined by the greedy station algorithm that were used in our experiments

approach, for each 15 minute interval of the week, we count the number of requests going from every origin region to every destination region. With this frequency table we are able to determine the probability of a given destination region given the origin region, time interval, and day of the week.

3.1.1 Discretization into regions

Given a list C of all the intersections in the road network of the city, we compute the set of regions such that in the resulting list no two centers are within a given radius, r, of each other, i.e. $\forall i, j \in C, ||i - j|| > r$. We employ Algorithm 1, where BallTree is a space partitioning data structure that allows for fast radius bounded nearest neighbor lookup. The data structure has query function, Query(c, r), that lets us find all the points within r of a point c. In Fig. 3-1 we show the centers of the regions from a discretization of Manhattan, where a radius of 150 meters was used.

Algorithm 1 Pruning candidate region centers

1: $\mathcal{T} \leftarrow$ BallTree(C)
2: **for** $c \in C$ **do**
3: $\quad C \leftarrow C \setminus \mathcal{T}$.Query$(c, r)$

In Algo. 1 we are greedily pruning candidate region centers that are too close to our currently selected region center. This allows us to simply iterate through the candidates once.

3.1.2 Probability distribution

Given the set of region centers, we construct a probability distribution, $P(d \mid p; \xi, w)$, which is the probability of destination region d, given the origin region p, time interval ξ, and day of the week w. We partition each day into 15 minute intervals resulting in $1 \leq \xi \leq 96$.

This probability distribution can be generated via a frequentist approach. We used one year of historical taxi data consisting of 165,114,362 trips [1]. Each trip contained the origin and destination coordinates along with the time and date of the pick up. Using this data, we were able to populate a $96 \times 7 \times |C| \times |C|$ table, $\mathcal{F}$, indexed by the time interval, day of the week, origin region, and destination region with the number of times a given trip occurred. This allows us to determine the probability of a destination given the origin and a time period. The time period is defined as an initial and final time interval and the day of the week, $I = (\xi_0, \xi_1, w)$, resulting in the probability distribution of origin-destination

$$P(d, p \mid I) = P(p \mid I) \cdot P(d \mid p, I) \tag{3.1}$$

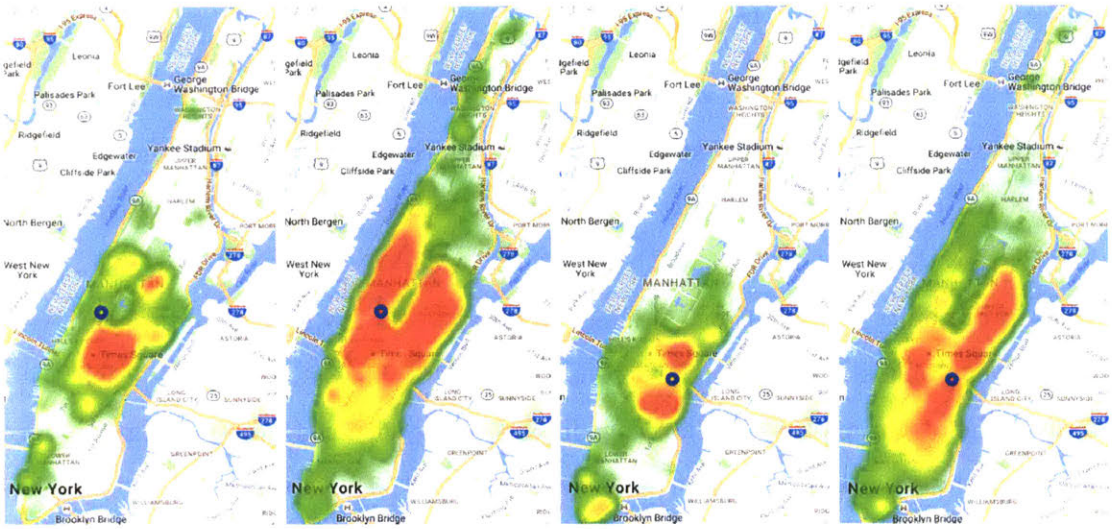
where

$$P(p \mid I, w) = \frac{\sum_{\xi=\xi_0}^{\xi_1} \sum_{i=1}^{|C|} \mathcal{F}_{\xi,w,p,i}}{\sum_{\xi=\xi_0}^{\xi_1} \sum_{i=1}^{|C|} \sum_{j=1}^{|C|} \mathcal{F}_{\xi,w,i,j}} \tag{3.2}$$

and

$$P(d \mid p, I, w) = \frac{\sum_{\xi=\xi_0}^{\xi_1} \mathcal{F}_{\xi,w,p,d}}{\sum_{\xi=\xi_0}^{\xi_1} \sum_{i=1}^{|C|} \mathcal{F}_{\xi,w,p,i}} \tag{3.3}$$

In Fig. 3-2 we show an example of the predicted demand for two fixed origins and two different time periods. Green indicates a lower probability and red indicates a higher probability. The first two heatmaps show the probability distributions of destinations given that the pick up location si on the west side of central park depicted by the blue dot. The left image shows the distribution from 7:30 to 8:30 and the right image shows the distribution from 21:30 to 22:00. The second two images show the



(a) Westside, 7:30 – 8:00 (b) Westside, 21:30 – 22:00 (c) Eastside, 7:30 – 8:00 (d) Eastside, 21:30 – 22:00

Figure 3-2: Heatmaps depicting the destination demand distribution. For this example, two locations in Manhattan are used as origins with a 30 minute interval to show the distribution. For each location, two intervals are used to show different snapshots of the demand throughout the day.

heatmaps for the same time intervals given that the origin was to east of central park depicted by the blue dot. From this probability distribution we can sample future requests to anticipate demand.

3.1.3 Sampling of future demand

Consider a given period of time (ξ_0, ξ_1) and day of the week, w. We construct a list S, consisting of the cumulative sum of frequencies from the start time to the end time and another list L, of the same size, consisting of the corresponding origin-destination pairs. To sample requests we then generate a random number s, from 0 to $\max(S)$ and determine the index i of S such that $S_{i-1} \leq s \leq S_i$. We then return L_i which is the corresponding origin-destination pair of the cumulative sum of frequencies interval. This process, see Algorithm 2, allows us to draw samples from $\mathcal{D}(I)$. The function $\textsc{Rand}(0, N)$ returns a uniformly distributed random number from 0 to N. $\textsc{FindInterval}(S, s)$ returns the index i such that $S_{i-1} \leq s \leq S_i$ if $s > S_0$, otherwise it returns 0. This is done using binary search since S is sorted.

Algorithm 2 Sampling origin-destination pairs over time

1: $S \leftarrow \{\}, L \leftarrow \{\}$
2: **for** $\xi \in [\xi_0, \xi_1]$ **do**
3: **for** $(p, d) \in [1, |C|]^2$ **do**
4: $S \leftarrow S \cup \{\max(S) + \mathcal{F}_{\xi,w,p,d}\}$
5: $L \leftarrow L \cup \{(p, d)\}$
6: $s \leftarrow$ RAND$(0, \max(S))$
7: $i \leftarrow$ FINDINTERVAL(S, s)
8: **return** L_i

Lines 2 through 5 in Algo. 2 are iterating through all of the time interval and origin-destination pairs. For each time interval, origin, and destination for a given day of the week, we cumulatively sum the probabilities and store the corresponding origin-destination pair for that step of the sum. In line 6 we select a random number between 0 and sum of the probabilities and in line 7 we find corresponding index of the interval in the cumulative sum containing the random number. We then return the origin-destination pair from our stored list at that index.

Chapter 4

Passenger Assignment and Vehicle Routing

In this chapter we describe the algorithm developed for passenger assignment and vehicle routing. Specifically we develop the method for assigning vehicles to possible trips by computing which trips can be shared, determining which vehicles can service these shared trips, and solving an Integer Linear Program to compute an optimal assignment that takes into account predicted future requests. We also describe our method for rebalancing the remaining idle vehicles.

The chapter is structured as follows. Sec. 4.1 introduces the vehicle assignment problem and outlines our method to solve it. Sec. 4.2 describes the proposed approach in detail with Sec. 4.2.1 describing how we incorporate sampled future demand into the assignment and Sec. 4.2.2 formulating the program as an Integer Linear Program. Sec. 4.2.3 and 4.2.4 describe the pairwise graph of vehicle and requests (RV-graph) and the graph of candidate trips and pick-ups (RTV-graph) respectively which are two graphical structures used for optimization.

4.1 Preliminaries

In this section we introduce the notation employed throughout this thesis, followed by the problem formulation and an overview of the method.

4.1.1 Definitions

We consider a fleet $\mathcal{V}$ of m vehicles of capacity ν, the maximum number of passengers each vehicle can have at any given time. Denote the set of vehicles $\mathcal{V} = \{v_1, \ldots, v_m\}$. The current state of a *vehicle* v is given by a tuple $\{q_v, t_v, \mathcal{P}_v\}$, indicating its current position q_v, the current time t_v and its passengers $\mathcal{P}_v = \{p_1, \ldots, p_{n_v^{pass}}\}$. A *passenger* p is a request that has been picked-up by a vehicle.

We also consider a set of requests $\mathcal{R} = \{r_1, \ldots, r_n\}$. Where each travel request consists of the time of request, a pick-up location and a drop-off location. Formally, a *request* r is defined by a tuple $\{o_r, d_r, t_r^r, t_r^{pl}, t_r^p, t_r^d, t_r^*\}$, indicating its origin o_r, its destination d_r, the time of the request t_r^r, the latest acceptable pick-up time t_r^{pl} (initially given by $t_r^{pl} = t_r^r + \Omega$ with Ω the *maximum waiting* time), the pick-up time t_r^p, the expected drop off time t_r^d, and the earliest possible time at which the destination could be reached $t_r^* = t_r^r + \tau(o_r, d_r)$.

Given a graph of the streets with estimated travel times, a function $\tau(q_1, q_2)$ computes the travel time from q_1 to q_2, two positions in space encoded by their latitude and longitude coordinates. When a network representation of the map is available, standard techniques for efficiently computing shortest paths can be used [15].

We further define a *trip* $T = \{r_1, \ldots, r_{n_T}\}$ as a set of requests that can be combined and served by a single vehicle. A trip may have one or more candidate vehicles for execution and contain more requests than the capacity of the vehicle if they are picked and dropped of in a way that the capacity limit is satisfied at all times.

4.1.2 Problem formulation

We define the following problem.

Problem 1 (Informed batch assignment). *Consider a set of requests $\mathcal{R}$, a set of vehicles $\mathcal{V}$ at their current state including passengers, and a function to compute travel times on the road network. Compute the optimal assignment Σ of requests to vehicles that satisfies a set of constraints $\mathcal{Z}$, including a maximum capacity ν of passengers per vehicle, and that minimizes a cost function $\mathcal{C} = \mathcal{C}_{now} + \mathcal{C}_{future}$, where*

C_{now} could be the sum of travel delays for the current passengers and requests and C_{future} is a term which includes the cost of satisfying future predicted travel requests.

Our formulation is flexible with respect to physical and performance-related constraints $\mathcal{Z}$. In our implementation we consider the following ones:

- For each request r, the waiting time ω_r, given by the difference between the pick-up time t_r^p and the request time t_r^r, must be below a maximum waiting time Ω, for example 5 minutes.
- For each request r (or passenger p) the total travel delay $\delta_r = t_r^d - t_r^*$ ($\delta_p = t_p^d - t_p^*$) must be lower than a maximum travel delay Δ, for example 10 minutes, where t_r^d is the drop-off time and $t_r^* = t_r^r + \tau(o_r, d_r)$ the earliest possible time at which the destination could be reached if the shortest path between the origin o_r and the destination d_r was followed without any waiting time. The total travel delay δ_r includes both the in-vehicle delay and the waiting time.
- For each vehicle v, a maximum number of passengers, $n_v^{pass} \leq \nu$, for example capacity ten.

Ideally, all the requests shall be assigned to a vehicle, but given the constraints, this might not always be the case. Denote by $\mathcal{R}_{ok}$ the set of requests assigned to a vehicle and $\mathcal{R}_{ko}$ the set of requests that are not served by any vehicle.

Following [5], we define the cost C_{now} of an assignment Σ as the sum of travel delay over all passengers $\mathcal{P}$ and all assigned requests plus a large enough cost c_{ko} for each non-assigned request. Formally,

$$C_{now}(\Sigma) = \sum_{p \in \mathcal{P}} (t_p^d - t_p^*) + \sum_{r \in \mathcal{R}_{ok}} (t_r^d - t_r^*) + \sum_{r \in \mathcal{R}_{ko}} c_{ko} \tag{4.1}$$

To account for the future performance of the system, we introduce a new term C_{future}, which is the expected cost of serving future requests. This cost term is based on the predicted future demand with the objective of achieving a better routing

and assignment of the fleet towards the future requests. This will be discussed in Section 4.2.

For real-time fleet management, the method can be applied to continuous discovery and assignment of incoming requests. The proposed approach is to perform batch assignment of the requests within a short time span, for example every 30 seconds, to the fleet of vehicles. Problem 1 is invoked with the predicted state of the fleet at the assignment time and the cumulated requests. Requests that have not been picked-up by a vehicle within the previous assignment round are kept in the pool for assignment.

4.1.3 Method overview

The first step of the method consist on estimating, for each time of the day and for day of the week the amount of requests from each origin in the city to each destination. This is a probability distribution that is computed from historical data. We describe this step in Chapter. 3.

The main step of the method consists on solving Problem 1. To do so, at each assignment round we sample future requests from the estimated probability distribution and introduce them in the assignment and routing problem, albeit with lower cost that the real requests. This is described in Sec. 4.2. Fig. 4-1 shows a schema with the steps of the method, which we describe in the following.

The assignment and routing method is inspired by [5] and consists of the following four steps.

- Computing a pair-wise request-vehicle shareability graph (RV-graph). In this graph, requests r, predicted requests r^{pred} and vehicles v are pairwise connected if r, or r^{pred}, can be satisfied by v within the defined constraints and given the current state of v.

- Computing a graph (RTV-graph) of feasible trips (formed by one or more requests and/or predicted requests) and the vehicles that can serve them within the specified constraints.

- Solving an Integer Linear Program (ILP) to compute the best assignment of vehicles to trips.

- Rebalancing the remaining idle vehicles towards areas with a deficit of vehicles and too many requests via a Linear Program (LP).

Given that the problem at hand is NP hard, obtaining an optimal assignment can be computationally expensive. For practical applications it is required that a sub-optimal solution is returned within an allocated runtime budget, which might be improved incrementally up to optimality. The proposed algorithm does present this anytime-optimal property.

An example of the method is shown in Fig. 4-1. The first image (a) shows an example of a street network with two requests (orange human = origin, red triangle = destination), two predicted requests (blue human = origin, red triangle = destination) and two vehicles (yellow car = origin, red triangle = destination of passenger). Vehicle 1 has one passenger and vehicle 2 is empty. With the passengers origin and destinations, a pairwise shareability RV-graph of requests and vehicles is constructed shown in (b). Cliques of this graph are potential trips. The image in (c) shows RTV-graph of candidate trips and vehicles which can execute them. A node (yellow triangle) is added for requests that can not be satisfied. The optimal assignment given by the solution of the ILP, where vehicle 1 serves requests 2 and 3 and vehicle 2 serves requests 1 and 4 is then computed. This is shown in (d). Using the assignment, the vehicles executed their planned routes shown in (e). The predicted requests alter the route of the vehicles, driving them towards areas of likely future requests.

4.2 Method for routing and assignment

The goal is to bias the vehicles towards areas where future requests are more likely to appear. The method takes into account the current state of the fleet, the current set of requests, as well as the predicted demand, consisting of both origins and destinations. The method computes a batch assignment of the current requests in the requests pool

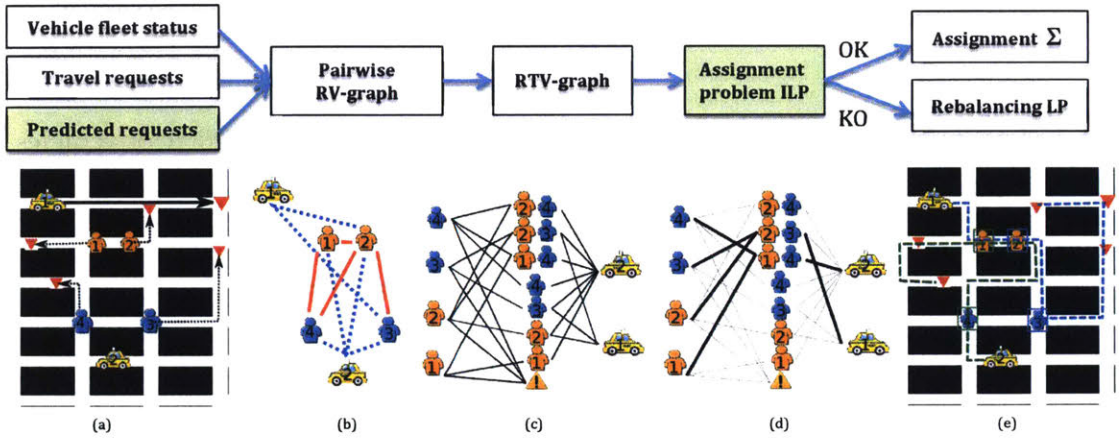


Figure 4-1: Schematic overview of the proposed method for batch assignment of multiple requests to multiple vehicles of capacity ν. The method consists of several steps leading to an integer linear optimization which provides an anytime optimal assignment.

$\mathcal{R}$ to the vehicles of the fleet $\mathcal{V}$. For real scenarios with incoming requests, this routing and assignment is performed at a constant frequency, which in our experiments was once every 30 seconds. Fig. 4-1 shows a schema with the steps of the method, which we describe in the following.

4.2.1 Sampling of future requests

In each iteration of the batch optimizer, a set of additional requests $\mathcal{R}_{future}$ are sampled from a historical probability distribution of future demand with the method described in Sec. 3.1.3. We first define a time interval for the predictions $I_{pred} = [t_{now}, t_{pred}, w]$, where t_{now} is the current time and t_{pred} a time in the future, which in our experiments is set to $t_{now} + 1800s$ for an interval of 30 minutes in the future, and w is the day of the week. We also define a maximum number of samples n_{pred}^{max}.

At run time, the number of samples is given by

$$n_{pred} = \min(n_{pred}^{max},\ E(\mathcal{D}(I_{pred}))), \tag{4.2}$$

where $E(\mathcal{D}(I_{pred}))$ denotes the number of expected requests in interval I_{pred}, given the distribution $\mathcal{D}$ estimated in Chapter 3.

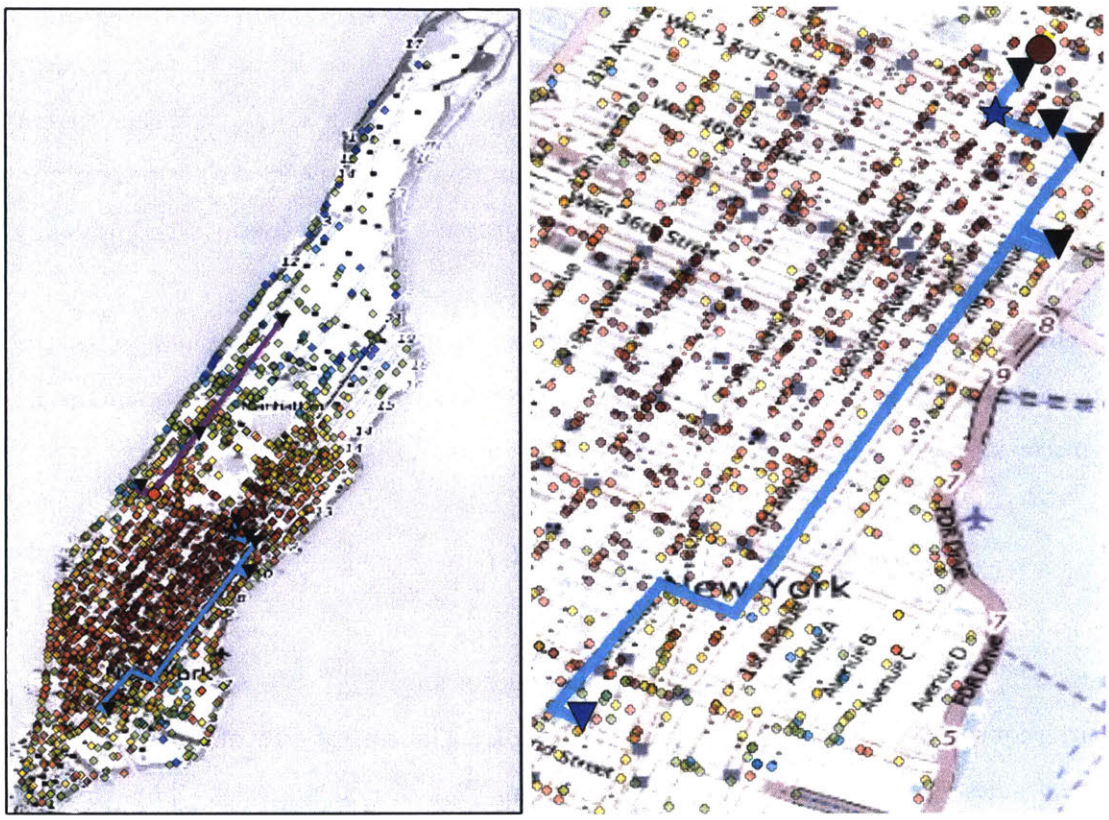


Figure 4-2: Left: Snapshot, 2000 vehicles, capacity 4. $\omega = 5$ min, Wednesday 8pm. Vehicle fleet represented at their current positions. Colors indicate number of passengers (0: light blue, 1: light green, 2: yellow, 3: dark orange, 4: dark red). 39 rebalancing vehicles are displayed in dark blue - mostly in the upper Manhattan returning to the middle. Right: Close view of the scheduled path for a vehicle (dark-red circle) with four passengers, which drops one off, pick-up a new one (blue star) and drops all four. Drop-off locations are displayed with inverted triangles.

Each future request $r_i^{pred} \in \mathcal{R}_{future}$ is sampled, via Algorithm 2, from $\mathcal{D}$ and the time interval,

$$r_i^{pred} \sim \mathcal{D}(I_{pred}). \tag{4.3}$$

At each time step, after each batch assignment, the set $\mathcal{R}_{future}$ is cleared. New future requests will be sampled in the following time step, every 30 seconds in our experiments.

4.2.2 Optimization

These requests are added to the pool of requests $\mathcal{R}^+ := \mathcal{R} + \mathcal{R}_{future}$ for the current iteration (and removed afterwards). Vehicles can then be matched with trips containing future requests in $\mathcal{R}_{future}$ and may make progress towards them (although they can not be picked since they are virtual).

The additional requests $\mathcal{R}_{future}$ are subject to the same constraints $\mathcal{Z}$ as the real requests $\mathcal{R}$, and enter the assignment problem via the additional term in the optimization cost $\mathcal{C}_{future}$. Following Eq. (4.1), this term is defined as

$$\mathcal{C}_{future}(\Sigma) = \sum_{r \in \mathcal{R}_{ok}^{pred}} (t_r^d - t_r^*) + \sum_{r \in \mathcal{R}_{ko}^{pred}} c_{ko}^{pred}, \tag{4.4}$$

where $\mathcal{R}_{ok}^{pred}$ is the set of assigned future requests and $\mathcal{R}_{ko}^{pred}$ the set of unassigned future requests, such that $\mathcal{R}_{ok}^{pred} \cup \mathcal{R}_{ko}^{pred} = \mathcal{R}_{future}$. The cost of a future request being ignored satisfies $c_{ko}^{pred} << c_{ko}$, much lower than that of real requests. This process gives preference to real requests, with a bias in the assignment and routing towards servicing areas of higher expected future demand.

Following Sec. 4.1.3 the batch assignment algorithm consists of the following steps:

- Sample a set of requests $\mathcal{R}_{future} \sim \mathcal{D}$.
- Compute a pair-wise request-vehicle shareability graph (RV-graph) between the requests $\mathcal{R}^+$ and the vehicles $\mathcal{V}$. In this graph, request r and vehicle v are connected if, given the current state of v, request r can be satisfied by v while respecting the defined constraints $\mathcal{Z}$ for maximum waiting time, delay and vehicle capacity.
- Compute a graph (RTV-graph) of feasible trips (formed by one or more requests) and the vehicles that can serve them within the specified constraints. Each trip may contain both real and predicted requests. Feasible trips are computed incrementally for each vehicle. Each trip is linked in the graph to the requests that form it and the vehicles that can serve it while respecting the constraints

$\mathcal{Z}$.

- Compute a greedy assignment Σ_{greedy}, where trips are assigned to vehicles iteratively in decreasing size of the trip and increasing cost. The idea is the maximize the amount of requests served while minimizing cost.

- Starting from the greedy assignment solve an Integer Linear Program to compute an optimal assignment Σ_{optim} of vehicles to trips, and therefore to requests. Following [5], a binary variable is added for each link between a feasible trip and a vehicle that can execute it within the RTV-graph. This assignment also provides the optimal routes, as computed in the RTV-graph.

- Rebalance the remaining idle vehicles towards areas with a deficit of vehicles and too many requests via a Linear Program. The idle vehicles are assigned to the unassigned requests of the previous step.

Following the notation of [5], the new Integer Linear Program (fifth step of the method, see Algorithm 3) consists of the following binary variables

$$\mathcal{X} = \{\epsilon_{i,j},\ \chi_k;\ \forall (T_i, v_j) \text{ edge in RTV-graph}, \forall r_k \in \mathcal{R}^+\}.$$

From Eq. (4.1) and Eq. (4.4) the cost terms $c_{i,j}$ are given by the sum of delays for all the passengers and requests associated to a trip T_i, as served by a vehicle v_j

$$c_{i,j} = \sum_{r \in \mathcal{I}^R_{T=i,V=j}} (t^d_r - t^*_r) + \sum_{p \in \mathcal{I}^P_{V=j}} (t^d_p - t^*_p), \tag{4.5}$$

where $\mathcal{I}^R_{T=i,V=j}$ denotes the requests in trip T_i as served by vehicle v_j, and $\mathcal{I}^P_{V=j}$ the passengers of vehicle v_j.

The optimal assignment is obtained by solving the ILP of Algorithm 3. Recall that: $\mathcal{E}_{TV}$ denotes the edges between a trip T_i and a vehicle v_j in the RTV-graph (i.e. there exists a route for which vehicle v_j can serve trip T_i within the given constraints $\mathcal{Z}$); $\mathcal{I}^T_{V=j}$ denotes the trips that can be served by vehicle v_j; $\mathcal{I}^T_{R=k}$ denotes the trips

Algorithm 3 Optimal assignment

1: Initial guess: Σ_{greedy}

2: $\Sigma_{optim} := \arg\min_{\mathcal{X}} \sum_{i,j \in \mathcal{E}_{TV}} c_{i,j}\epsilon_{i,j} +$

3: $\quad + \sum_{1 \leq k \leq n} c_{ko}\chi_k + \sum_{n+1 \leq k \leq n+n_{pred}} c^{pred}_{ko}\chi_k$

4: $\quad$ s.t. $\sum_{i \in \mathcal{I}^T_{V=j}} \epsilon_{i,j} \leq 1 \qquad \forall v_j \in \mathcal{V}$

5: $\quad \sum_{i \in \mathcal{I}^T_{R=k}} \sum_{j \in \mathcal{I}^V_{T=i}} \epsilon_{i,j} + \chi_k = 1 \quad \forall r_k \in \mathcal{R}^+$

(combinations of requests) in which request r_k can be served; and $\mathcal{I}^V_{T=i}$ denotes the vehicles that can serve trip T_i.

After assignment and routing, the vehicles make progress towards their assigned requests, picking requests (which become passengers) as they reach them, and the set $\mathcal{R}_{future}$ is cleared. Then, this process is repeated at the desired frequency with the incoming requests. Fig. 4-2 shows a snapshot of the algorithm being executed with the vehicle making progress towards its assigned requests.

4.2.3 Pairwise Graph of Vehicles and Requests (RV-graph)

The first step of the method is to compute (a) which requests can be pairwise combined, and (b) which vehicles can serve which requests individually, given their current passengers. This step builds on the idea of share-ability graphs proposed by [37], but it is not limited to the requests and includes the vehicles at their current state as well.

Two requests r_1 and r_2 are connected in the graph if they can potentially be combined. This is, if a virtual vehicle starting at the origin of one of them could pick-up and drop-off both requests while satisfying the constraints $\mathcal{Z}$ of maximum waiting time and delay. A cost $\sum_{r=\{1,2\}} (t^d_r - t^*_r)$ can be associated to each edge $e(r_1, r_2)$.

Likewise, a request r and a vehicle v are connected if the request can be served by the vehicle while satisfying the constraints $\mathcal{Z}$. This is, if $travel(v, r)$ returns a valid trip that drops the current passengers of the vehicle and the picked request r within the specified maximum waiting and delay times. The edge is denoted by $e(r, v)$.

Limits on the maximum number of edges per node can be imposed, trading-off optimality at the later stages. Speed-ups such as the ones proposed in T-share [27] could be employed in this stage to prune the most likely vehicles to pick up a request.

This graph, denoted *RV-graph*, gives an overview of which requests and vehicles might be shared. In Figure 4-3 an example of the RV-graph is shown with 90 requests and 30 vehicles.

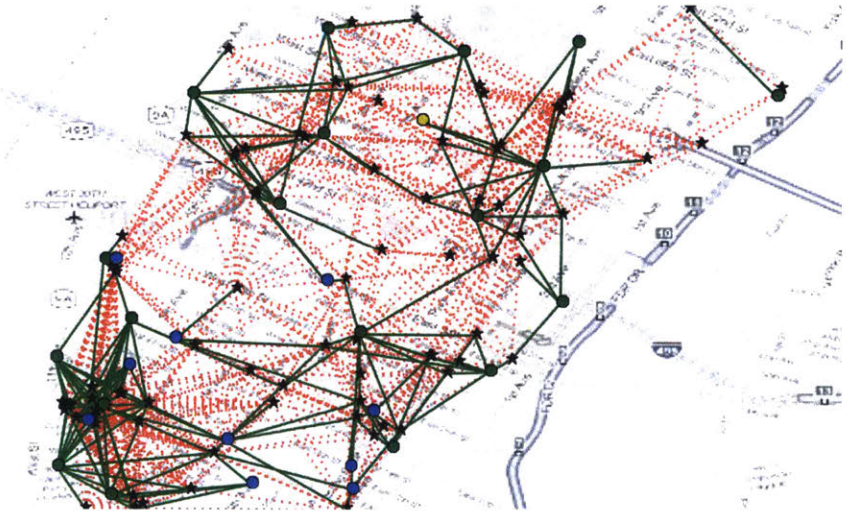


Figure 4-3: Example of a pairwise RV-graph for 90 requests (star) and 30 vehicles (circle) with edges between two requests in dotted red and between a request and a vehicle in solid green. The maximum waiting time and delay are three and six minutes in this example.

4.2.4 Graph of Candidate Trips and Pick-ups (RTV-graph)

The second step of the method is to explore the regions of the RV-graph for which its induced subgraph is complete, or cliques, to find feasible trips. Recall that a trip T is defined by a set of requests $T = \{r_1, \ldots, r_{n_T}\}$. A trip is feasible if the requests can be combined, picked-up and dropped-off by some vehicle, while satisfying the constraints $\mathcal{Z}$.

A request may form part of several feasible trips of varying size, and a trip might admit several different vehicles for execution. The request-trip-vehicle RTV-graph contains edges $e(r, T)$, between a request r and a trip T and feasible edges $e(T, v)$,

between a trip T and a vehicle v. Namely,

$$\exists\, e(r,T) \Leftrightarrow r \in T \tag{4.6}$$

$$\exists\, e(T,v) \Leftrightarrow travel(v,T) = \text{"valid"} \tag{4.7}$$

The algorithm to compute the feasible trips and edges proceeds incrementally in trip size for each vehicle, as shown in Algorithm 4, where $\mathcal{T}$ is the list of feasible trips. With each node $e(T,v)$, the cost $\mathcal{C}$ of the trip and pick-up is stored. For each vehicle, a timeout can be set after which no more trips are explored. This leads to suboptimality of the solution, but faster computation, removing longer trips.

Algorithm 4 Generation of RTV-graph

1: $\mathcal{T} = \emptyset$
2: **for** each vehicle $v \in \mathcal{V}$ **do**
3: $\quad \mathcal{T}_k = \emptyset\ \forall k \in \{1,\dots,\nu\}$
4: $\quad$ [Add trips of size one]
5: $\quad$ **for** $e(r,v)$ edge of RV-graph **do**
6: $\quad\quad \mathcal{T}_1 \leftarrow T = \{r\}$; Add $e(r,T)$ and $e(T,v)$
7: $\quad$ [Add trips of size two]
8: $\quad$ **for** all $\{r_1\},\{r_2\} \in \mathcal{T}_1$ and $e(r_1,r_2) \in$ RV-graph **do**
9: $\quad\quad$ **if** $travel(v,\{r_1,r_2\}) =$ valid **then**
10: $\quad\quad\quad \mathcal{T}_2 \leftarrow T = \{r_1,r_2\}$; Add $e(r_i,T)$ and $e(T,v)$
11: $\quad$ [Add trips of size k]
12: $\quad$ **for** $k \in \{3,\dots,\nu\}$ **do**
13: $\quad\quad$ **for** all $T_1, T_2 \in \mathcal{T}_{k-1}$ with $|T_1 \cup T_2| = k$ **do**
14: $\quad\quad\quad$ Denote $T_1 \cup T_2 = \{r_1,...,r_k\}$
15: $\quad\quad\quad$ **if** $\forall i \in \{1,...,k\}$, $\{r_1,...,r_k\} \setminus r_i \in \mathcal{T}_{k-1}$ **then**
16: $\quad\quad\quad\quad$ **if** $travel(v, T_1 \cup T_2) =$ valid **then**
17: $\quad\quad\quad\quad\quad \mathcal{T}_k \leftarrow T = T_1 \cup T_2$
18: $\quad\quad\quad\quad\quad$ Add $e(r_i,T)$, $\forall r_i \in T$, and $e(T,v)$
19: $\quad \mathcal{T} \leftarrow \bigcup_{i\in\{1,\dots,\nu\}} \mathcal{T}_i$

Note that steps 7 and 12 of the Algorithm can be efficiently implemented by employing ordered lists with respect to the request ids. This step can be parallelized among the vehicles. Lines 5 and 6 adds trips of size one to the RTV-graph. Lines 8 through 10 add trips of size two and lines 12 through 18 add trips of arbitrary size up to the capacity of the vehicle.

4.2.5 Rebalancing

After the assignment, due to fleet imbalances, the set $\mathcal{R}_{ko}$ of unassigned requests may not be empty, and some empty vehicles $\mathcal{V}_{idle}$ may still by unassigned to any request. This may occur when the idle vehicles are in areas far away from the area of current request, and due to the maximum waiting time and delay constraints and vehicle capacity. Under the assumptions that, (a) ignored requests may wait longer and request again, (b) it is likely that more requests occur in the same area where all requests can not be satisfied and (c) there are not enough requests in the neighborhood of the idle cars, we propose the following approach to rebalance the fleet by moving only the idle vehicles.

To **rebalance** the vehicle fleet, after each batch assignment, the vehicles in $\mathcal{V}_{idle}$ are assigned to requests in $\mathcal{R}_{ko}$ to minimize the sum of travel times, with the constraint that either all requests or all the vehicles are assigned. For this, we first compute the travel time $\tau_{v,r}$ of each individual request r - vehicle v pickup and then obtain the optimum of the assignment via a fast Linear Program described in Algorithm 5. In this approach, if all requests can be satisfied some vehicles may remain idle, saving fuel and distance travelled, which is the case at night time.

Algorithm 5 Rebalancing

1: Given: the idle (empty, stopped and unassigned) vehicles $\mathcal{V}_{idle}$, and the unassigned requests $\mathcal{R}_{ko}$.
2: Given: the shortest travel time $\tau_{v,r}$ for vehicle $v \in \mathcal{V}_{idle}$ to pick request $r \in \mathcal{R}_{ko}$.
3: Variables: $\mathcal{Y} = \cup_{v \in \mathcal{V}_{idle}, r \in \mathcal{R}_{ko}} y_{v,r}$. Where $y_{v,r} \in \mathbb{R}$ indicates individual assignments.
4:
5: $\Sigma_{rebalance} := \arg\min_{\mathcal{Y}} \sum_{v \in \mathcal{V}_{idle}} \sum_{r \in \mathcal{R}_{ko}} \tau_{v,r}\ y_{v,r}$
6: s.t. $\sum_{v \in \mathcal{V}_{idle}} \sum_{\mathcal{R}_{ko}} y_{v,r} = \min(|\mathcal{V}_{idle}|, |\mathcal{R}_{ko}|)$
7: $0 \leq y_{v,r} \leq 1 \quad \forall y_{v,r} \in \mathcal{Y}$.
8:
9: Where $|.|$ denotes the number of elements of a set.
10: The solution of this Linear Program is also a solution of the Integer Linear Program with $y_{v,r} \in \{0, 1\}$.

In Algo. 5, the fast linear program for rebalance is formally described. In line 5, we are describe our objective function which is minimizing the travel time costs

for assigning an idle vehicle to an unassigned request. Line 6 shows our equality constraint for the total number of assignments which needs to be equal to lesser of the number of idle vehicles or number of unassigned requests. The resulting assignment will either assign all of the idle vehicles or all the unassigned requests.

4.3 Scope and Limitations

This chapter presented an method to automatically pool requests for mobility on demand into feasible trips and optimally assign vehicles to these trips. Even though in this thesis is focused on ride-sharing and mobility on demand systems, this method has a much larger scope and can be potentially applied to other problems in which you have a number of agents with capacity constraints servicing pick-up and delivery requests that could be shared and serviced by the same agent. Logistics and package delivery is a good example of one of these problems where you have trucks able to carry multiple packages to different delivery locations. Our methodology can also be applied to content delivery problems on the internet where you have groups of nodes serving content to clients. Here the passenger is analogous to the content, the capacity to the node bandwidth, origin location to the content host, and the delivery location to the client's computer.

Our approach is limited in the sense that we do not fully utilize the vehicle fleet even though there are requests that go unserviced. This is due to our maximum waiting time constraints that reduce the complexity of the problem by letting us compute which requests can be shared. To overcome this problem, we rebalance the vehicles in the network and predictively position vehicles to anticipate future demand. Unfortunately, rebalancing and predictive positioning are not wrapped up as part of the original optimization problem but come only when we have assigned the vehicles.

Chapter 5

Results

In this chapter, we examine the performance of the assignment algorithm with and without predictive positioning. We show that the algorithm is able to significantly reduce the amount of vehicles needed to service the demand in Manhattan. We also show that incorporating future requests into the assignment in order to anticipate future demand helps reduce the waiting time and delay experienced by passengers. The chapter is structured as follows. Sec. 5.1 describes the performance of the base algorithm without using predictive positioning. Sec. 5.1.1 analyzes the robustness of the algorithm to changes in the interval length for pooling and the density of demand. Sec. 5.2 examines the benefit of anticipating future demand by incorporating predictive positioning into the algorithm.

5.1 Experiments without predictive positioning

We assess the performance of a MoD fleet controller using the proposed algorithm, against real data from an arbitrarily chosen representative week, from 00:00 Sunday 5th May 2013 to 23:59 Saturday 11th May 2013, from the publicly available dataset of taxi trips in Manhattan, New York, USA [18]. This dataset contains for each day the time and location of all the pick-ups and drop-offs executed by each of the 13,586 active taxis. From this data we extract all the requests (origin and destination within Manhattan) and consider the time of request equal to the time of pick-up. We

consider the complete road network of Manhattan (4092 nodes and 9453 edges), with the travel time on each edge (road segment) of the network give by the daily mean travel time estimate, computed using the method in [37]. Shortest paths and travel times between all nodes are then precomputed and stored in a look-up table.

We perform a simulation of the evolution of the taxi fleet, where vehicles are initialized at midnight at sampled positions from a historical demand distribution and continuously travel to pick up and drop off passengers to satisfy the real requests extracted from the dataset. Requests are collected during a time window, 30 seconds in our experiments, after which they are assigned in batch to the different vehicles. Past requests are kept in the requests pool until picked-up and can be reassigned if a better match is found before pick-up. Each day contains between 382,779 (Sunday) and 460,700 (Friday) requests, and the running pool of requests contains up to 2,000 requests at any given time. The method is robust both with respect to the chosen time window and the density of demands, as shown in Sec. 5.1.1 in results with a time window between 10 and 50 seconds, and with half/double the amount of requests (~220,000/~880,000 per day) in NYC.

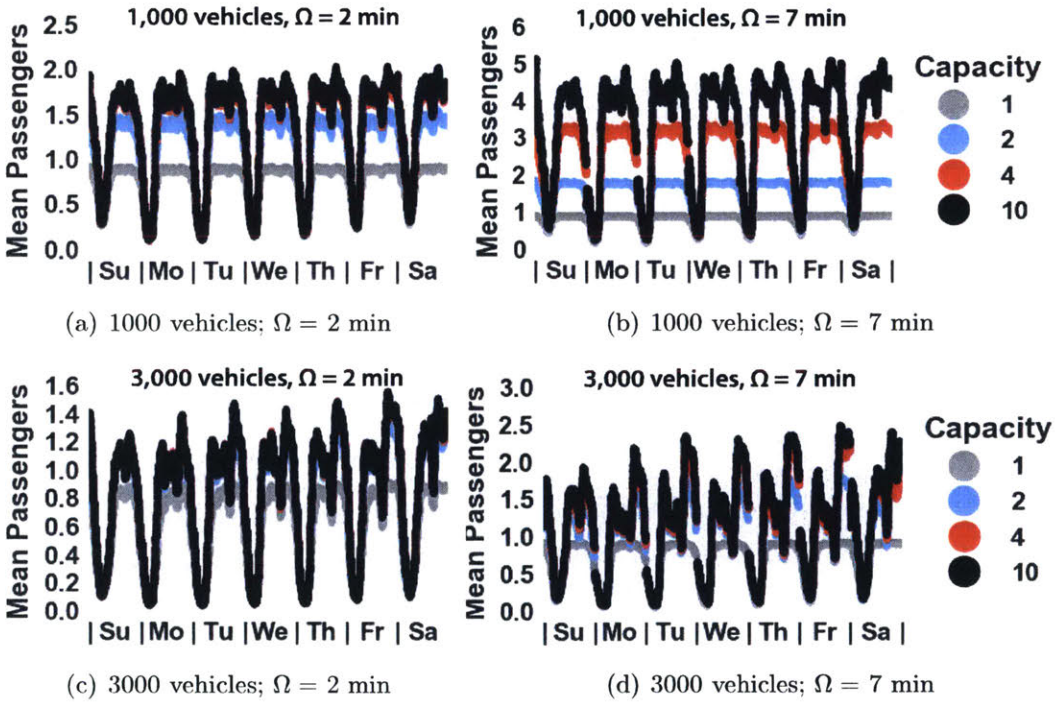
We analyze several metrics, with different vehicle fleet sizes ($m \in \{1000, 2000, 3000\}$ vehicles), vehicle capacities ($\chi \in \{1, 2, 4, 10\}$ passengers) and maximum waiting times ($\Omega \in \{120, 300, 420\}$ seconds). The maximum trip delay Δ is double the maximum waiting time and includes both the waiting time ω and the inside-the-vehicle travel delay. Our analysis shows that, thanks to high capacity ride sharing, a reduced fleet of vehicles (below 25% of the active taxis in NYC) is able to satisfy 99% of the requests with mean waiting time and delay of about 2.5 minutes. All results in this section include rebalancing of idle vehicles to unassigned requests; experimentally we observed that the rebalancing step contributed an increase in the service rate of about 20%, see Table 1 in the appendix.

A high resolution version of the accompanying video, which shows the evolution of the taxi fleet in NYC for a subset of experiments, is available at `https://youtu.be/EzOWFu7fMDO`.

High vehicle occupancy is achieved in times of high demand, with a large number

of the trips being shared. In Fig. 4-2 we observe that many vehicles are located in mid-Manhattan and contain three/four passengers. Fig. 5-1 shows that the occupancy depends on the fleet size, capacity and the maximum waiting/delay time. Lower fleet size, larger capacity and longer waiting/delay times increase the possibilities for ride sharing and lead to higher mean vehicle occupancy. In Fig. 5-2 we observe that during peak hours a small fleet of high capacity vehicles does indeed operate at high occupancy. For 1000 vehicles of capacity ten this is about 10% of the fleet with 8 or more passengers, 40% with 6 or more, 80% with 3 or more and 98% with 1 or more. For 2000 vehicles of capacity four, more than 70% of them have at least three passengers at 8pm.

We observe that the value of fleets with larger passenger capacities increases with larger Ω and Δ values, as expected, since passengers are willing to incur a larger personal time penalty. High capacity vehicles are also more important when the fleet



(a) 1000 vehicles; $\Omega = 2$ min

(b) 1000 vehicles; $\Omega = 7$ min

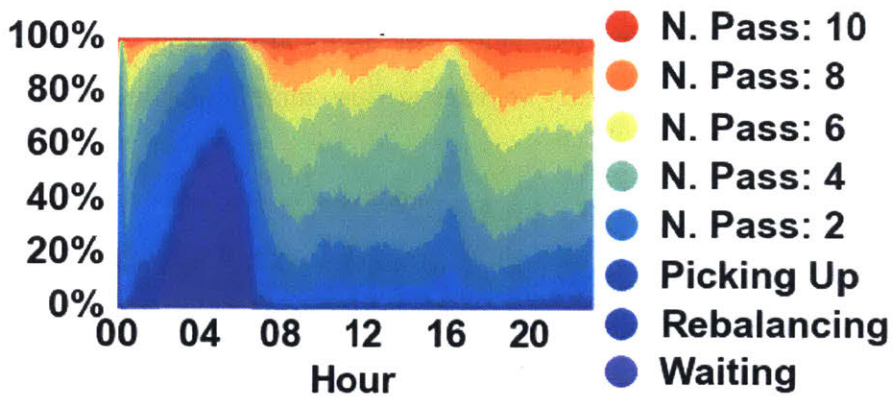
(c) 3000 vehicles; $\Omega = 2$ min

(d) 3000 vehicles; $\Omega = 7$ min

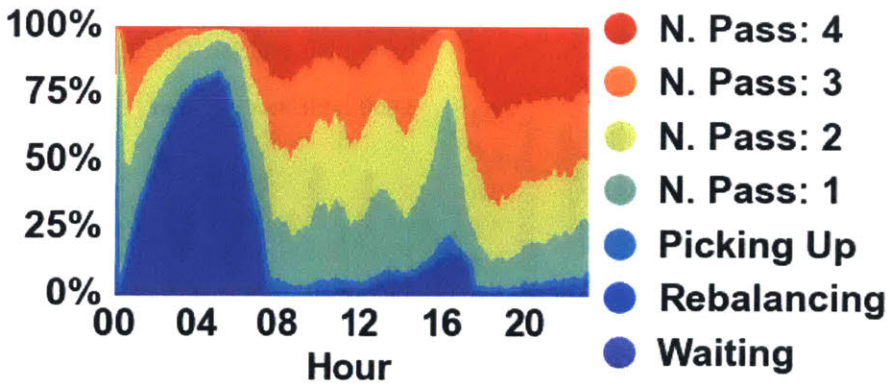
Figure 5-1: Mean number of passengers. Time series over a week for fleet sizes of 1000 and 3000 vehicles of varying capacity and maximum waiting time. At night most vehicles wait and during rush hour the mean occupancy decreases as the fleet gets larger. Larger maximum waiting time enables more opportunities for ride sharing.

size is smaller, as seating capacity might be a bottleneck with smaller fleets. For instance, see Fig. 5-5(a), a fleet of 1000 vehicles with capacity 10 can satisfy almost 80% of the requests with $\Omega = 420s$, compared to below 30% for a single rider taxi, for a net gain of over 50%. However. with a larger fleet of 3000 vehicles and $\Omega = 120s$, the benefit is only about 15%. Interestingly, if longer waiting times and delays are allowed, $\Omega = 420s$, a fleet of 3000 vehicles of capacity 2, 4 and 10 could serve 94%, 98% and 99% of the demand. To achieve 98% service rate, a fleet of just 2000 vehicles of capacity 10 was required. This represents a reduction of the fleet size to 15% of the active taxi fleet in NYC.

As expected, the in-car travel delay does increase with the increase in vehicle

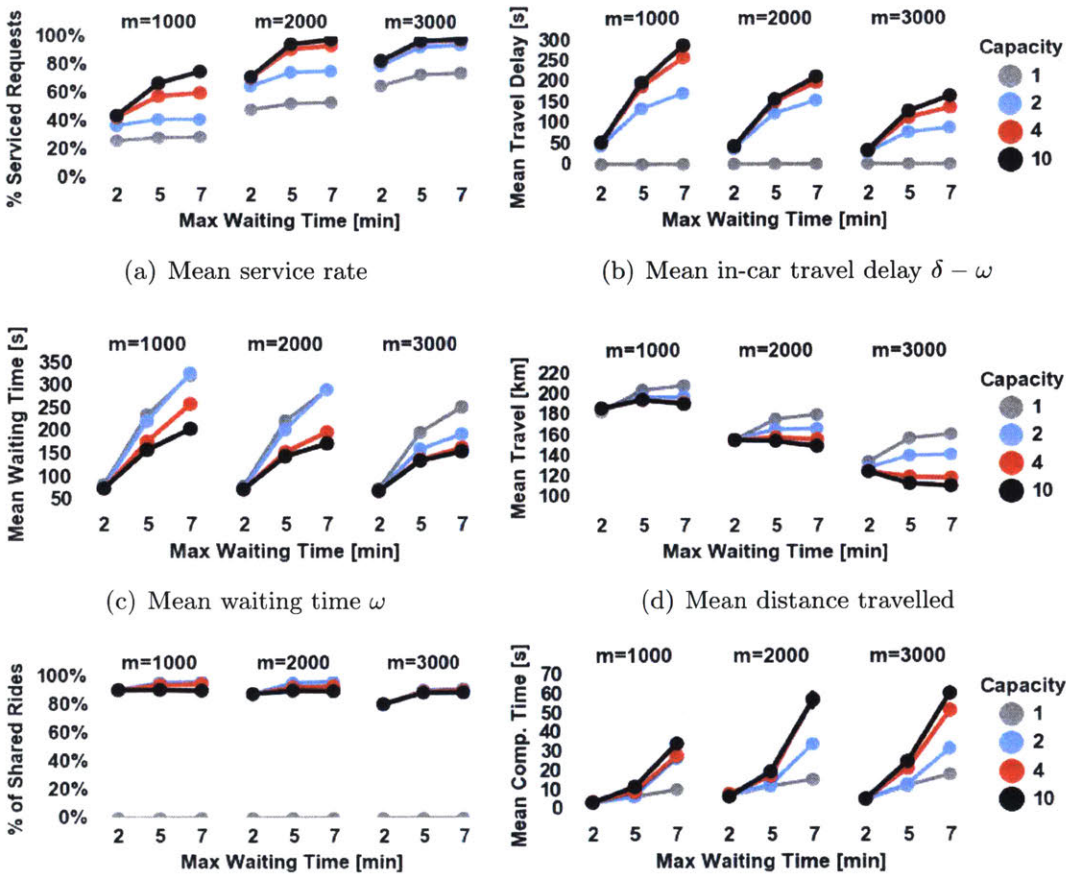


(a) Fleet of 1000 vehicles of capacity ten; $\Omega = 7$ min; Friday



(b) Fleet of 2000 vehicles of capacity four; $\Omega = 5$ min; Friday

Figure 5-2: Percentage of vehicles in each state (waiting, rebalancing and number of passengers) for a representative day (Friday 00h to 24h). (a) A fleet of 1000 vehicles of capacity ten with many opportunities for ride sharing in high capacity vehicles. (b) A fleet of 2000 vehicles of capacity four, showing the utility of full vehicle sharing.



(a) Mean service rate

(b) Mean in-car travel delay $\delta - \omega$

(c) Mean waiting time ω

(d) Mean distance travelled

(e) Percentage of shared rides

(f) Mean computational time

Figure 5-3: Comparison of several performance metrics for varying vehicle capacity (1, 2, 4 and 10 passenger, shown with lines). Each subplot is for a fleet size of 1000, 2000 and 3000 vehicles and the coordinate axis show increasing maximum waiting time Ω of 2, 5 and 7 minutes.

capacity, see Fig. 5-5(b). Nonetheless, that increase seems practically negligible - well below 100 s - once ride-sharing is allowed. Furthermore, the mean waiting time does in fact decrease as vehicle capacity is increased, see Fig. 5-5(c). For a fleet size of 1000 vehicles and $\Delta = 420s$, high capacity vehicles not only improved the service rate but also achieved a reduction in mean waiting time of over 100 s, which partially offsets the increased in-car delay. In particular, we observe that 3000 vehicles of capacity 2 and 4 could serve 94% and 98% of the demand with a mean waiting time of 3.2 and 2.7 minutes, and a mean delay of 1.5 min and 2.3 min, respectively. To achieve 98% service rate, with comparable waiting time (2.8 min) and delay (3.5 min)

a fleet of just 2000 vehicles of capacity 10 was required.

We also observed that increasing the vehicle capacity not only increases the service rate, but it also reduces the mean distance traveled by the vehicles in the fleet, see Fig. 5-5(d), potentially leading to a reduction in costs, congestion and pollution. Our online method results on about 90% shared rides, which slightly increases with Δ and decreases with the fleet size, see Fig. 5-5(e). Finally, we note that our approach is real-time capable, see Fig. 5-5(f). In our setup, for $\Omega \leq 300$s, the method is executed in less that 30s, which is the period for which requests are collected.

5.1.1 Robustness analysis

In this section we present results to confirm the robustness of the proposed method with respect to the length of the time window and the number - or density - of requests.

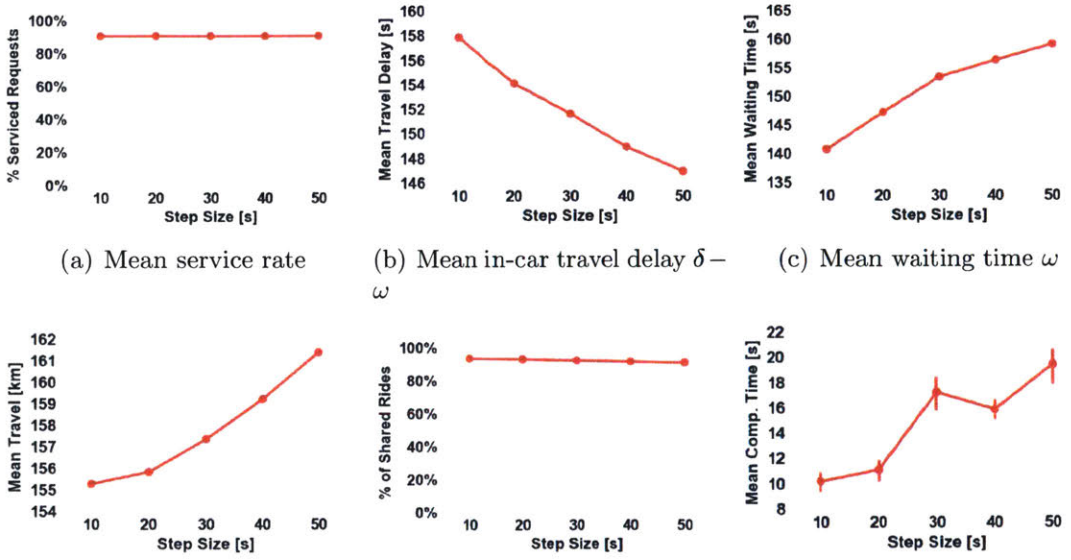
In each figure we analyze (a) service rate (percentage of requests serviced), (b) average in car delay $\delta - \omega$, (c) average waiting time ω, (d) average distance traveled by each vehicle during a single day, (e) percentage of shared rides (number of passengers who shared a ride, divided by the total number of picked-up passengers) and (f) average computational time for a 30 seconds iteration of the method, in a 24 core 2.5GHz machine, including computation of the RV-graph, computation of the RTV-graph, ILP assignment, rebalancing and data writing (higher levels of parallelization would drastically reduce this computational time).

Interval length

In Fig. 5-4 we show robustness results with respect to the interval length, this is, the period of time for which requests are aggregated before a new assignment to the fleet of vehicles. We compare different interval sizes of 10 s, 20 s, 30 s, 40 s and 50 seconds. Results are shown for a nominal case where we employ a fleet of 2000 vehicles of capacity 4 and a maximum waiting time Δ of 5 minutes. The points shown represent the average over a week of data in Manhattan with about 3 million requests.

In the experimental analysis of the effects of ride sharing shown in the previous section, we employed a time window of 30 seconds, which we considered reasonable when taking into account the computation cost of the approach and the time a person would be willing to wait for receiving an assignment.

In Fig. 5-4 we observe that the method is robust with respect to the time interval: the service rate and percentage of shared rides is mostly unchanged, the mean in-car travel delay slightly decreases with larger time intervals (better assignments are achieved), while both the mean waiting time and mean travelled distance by each vehicle do increase slightly with larger time intervals (the user have to wait longer to receive an assignment). The computational time of the method does increase with the size of the interval, since more requests are jointly assigned.



(a) Mean service rate (b) Mean in-car travel delay $\delta - \omega$ (c) Mean waiting time ω

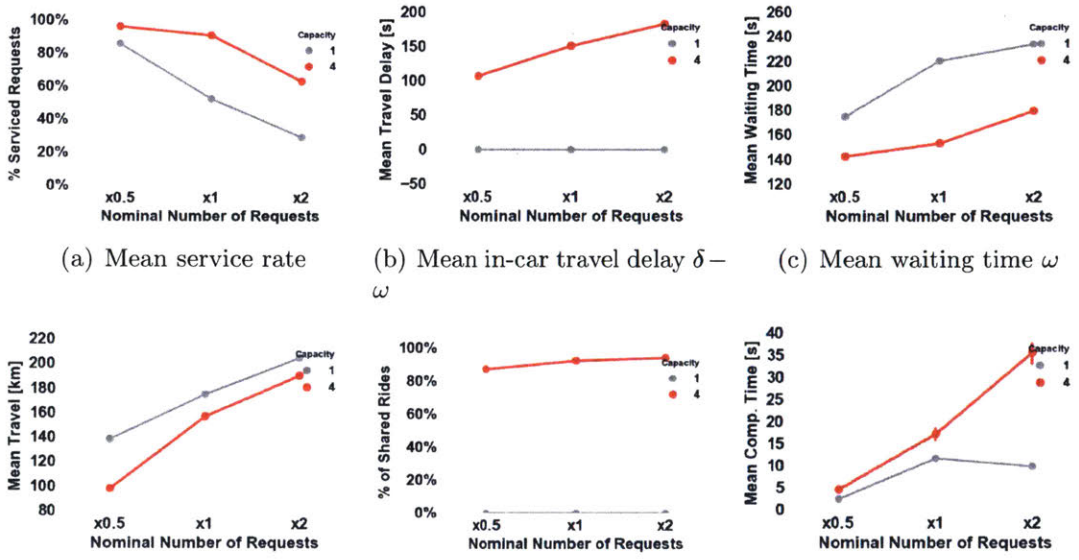
(d) Mean distance travelled (e) Percentage of shared rides (f) Mean computational time

Figure 5-4: Comparison of several performance metrics for varying interval size for the method, 10, 20, 30, 40 and 50 seconds. Results are shown for a nominal case where we employ a fleet of 2000 vehicles of capacity 4 and a maximum waiting time Δ of 5 minutes. The points shown represent the average over a week of data in Manhattan with about 3 million requests.

Density of demand

In Fig. 5-5 we show robustness results with respect to the density of demand, this is, the number of travel requests. We compare three different densities, the nominal one of Manhattan with about 3 million requests per week, half of the demand (x0.5) and double the demand (x2). To obtain half of the requests (x0.5) we sorted all the requests of each day by time and removed every odd line, this leads to about 1.5 million requests per week, or 200,000 per day. To obtain double the requests (x2), we cumulated the requests of the same day (e.g. Monday) for two consecutive weeks, this leads to about 6 million requests per week, or 850,000 per day. Results are shown for two nominal cases where we employ (i) a fleet of 2000 vehicles of capacity 4 and a maximum waiting time Δ of 5 minutes, and (ii) the same fleet but of capacity 1 (standard taxis). The points shown represent the average over a week of data in Manhattan.

We observe that the approach is robust with the decrease and increase in the number of requests, and that, as expected that performance metrics improve with a decrease in the number of requests.



(a) Mean service rate (b) Mean in-car travel delay $\delta - \omega$ (c) Mean waiting time ω

(d) Mean distance travelled (e) Percentage of shared rides (f) Mean computational time

Figure 5-5: Comparison of several performance metrics for varying density of requests. The nominal case [x1] is for a simulation with the real requests in Manhattan, of about 3 million per week. The case [x0.5] contains only half of the requests, about 1.5 million per week. And the case [x2] contains double the number of requests, about 6 million per week, or about 800,000 per day. Results are shown for two nominal cases where we employ (i) a fleet of 2000 vehicles of capacity 4 and a maximum waiting time Δ of 5 minutes, and (ii) the same fleet but of capacity 1 (standard taxis). The points shown represent the average over a week of data in Manhattan.

5.2 Experiments with predictive positioning

We assess the performance of the system with a fleet of 1000, 2000, and 3000 vehicles of capacity two and four passengers. We used a fixed maximum waiting time of $\Omega = 5$ minutes and a maximum delay of $\Delta = 10$ minutes. The minimum inter-station distance used for the region discretization was 150 meters. For the experiments, we use one week of historical taxi trip data from 00:00 on Sunday May 5th, 2013 to 23:59 on Saturday May 11th, 2013 to assess the performance of our algorithm. This data comes from a publicly available source of all taxi trips in Manhattan, New York, USA [18]. This dataset contains the geographical coordinates for the origins and destinations along with the associated pick up and drop off dates and times for all trips in executed by the 13,586 active taxis in New York City. From this data we consider the request and pick up time to be equal since the time for the request is not publicly available.

In order to find routes for the taxis to execute, we consider the entire road network of Manhattan. We estimate the travel time for each road segment using the daily mean travel time computed by the method in [37]. Different travel times were used for weekdays, Saturday, and Sunday. The shortest paths using these travel times were precomputed between every two intersections in the road network and were stored in a look-up table.

We initialize the vehicles each day at midnight at sampled positions from the demand distribution. We then simulate the execution of the fleet by issuing the requests obtained from the historical taxi dataset for the given day. The requests are collected within a 30 second time window after which they are assigned in batch to different vehicles using our algorithm of Sec. 4.2. In each time interval, or assignment step, we sample future requests up to 30 minutes in the future. We vary the number of predictions by using 0, 200, and 400 sampled predicted requests (per interval). These predicted requests enter the assignment problem of Algorithm 3, but are removed immediately afterwards, with new future requests being sampled in the following step. They do affect the assignment and routing at that time.

A pool of requests are kept until they have been picked up in case they can be reassigned to a better match. The number of requests in a single day varies from 382,779 on Sunday to 460,700 on Friday.

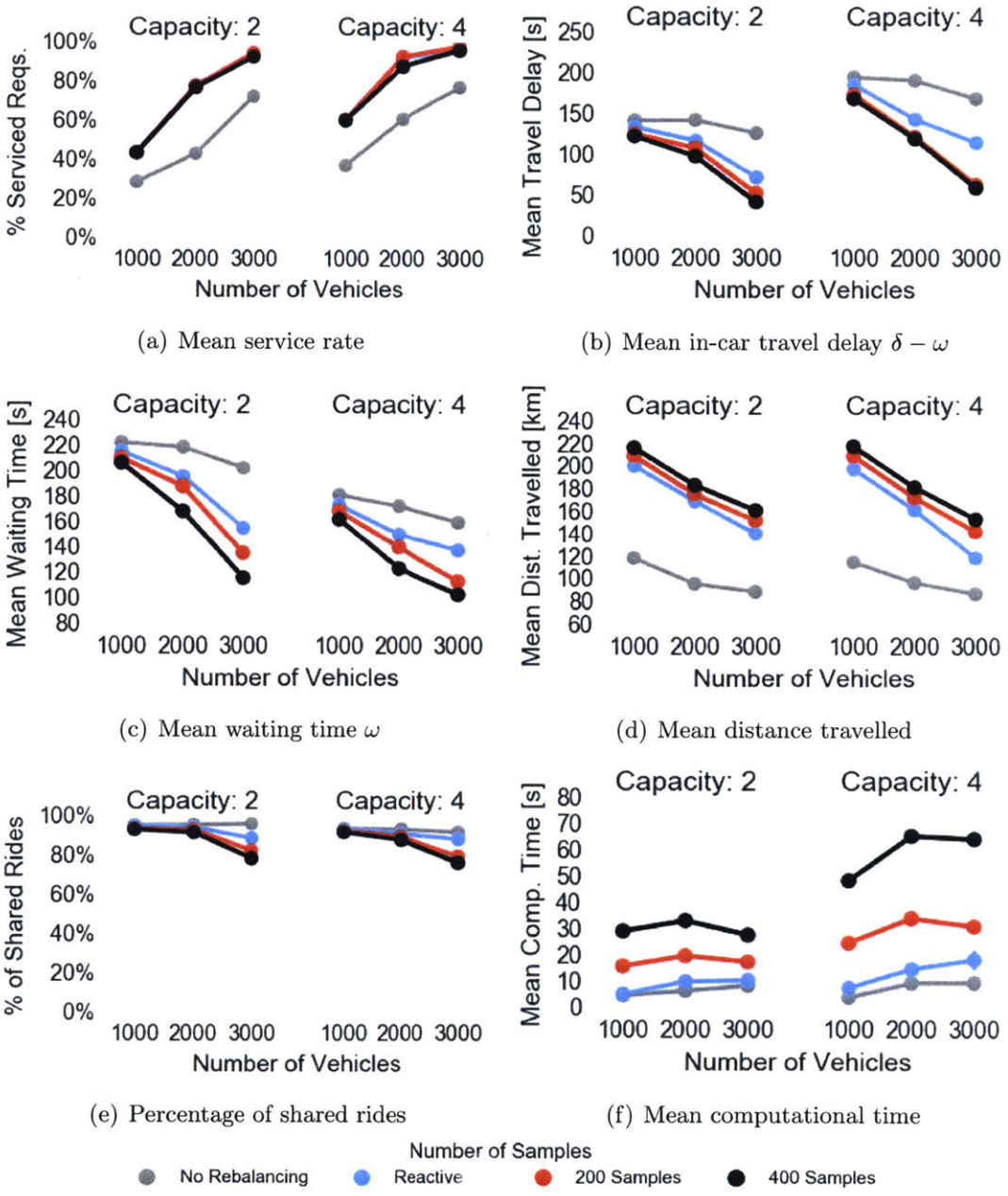
5.2.1 Results

We collect several metrics that characterize the system, including the service rate, in-car travel delay, waiting time, average distance traveled by the vehicles, percentage of shared rides, and the computational time. We use the same parameters as in [5], but with the additional sampled requests and cost term. These metrics are plotted for vehicle capacities two and four side by side in Fig. 5-6.

We observe that the service rate (number of requests serviced) remains approximately constant independently of the number of sampled requests, and it is close to 100% for 3,000 vehicles of capacity 4 (there are 13,000 active taxis per day in Manhattan). By sampling predicted requests we are able to reduce the mean in-car travel delay by 1.5 minutes and the mean waiting time by around 1 minute, with respect to the reactive approach.

Particularly, for the in-car travel delay and the waiting time, we see that there is a large benefit in using rebalancing and then a similar benefit by sampling predicted requests, see Fig.5-6-b) and -c). However, increasing the number of samples from 200 to 400 only marginally decreased the in-car travel delay by 3.4 seconds, when using a four passenger vehicle capacity and 3000 vehicles. It is likely that this small improvement is due to the time-outs introduced for real-time performance, which limit the benefit of additional samples. We believe that the increase would be larger if the algorithm was run to optimality.

We observe a trade-off between operational cost and performance, since the travel distance by the vehicles and the computational time of the approach do increase with the number of samples. The increase in travel distance arises from the fact that vehicles are routed towards predicted requests which may or may not appear in reality. This reduces mean waiting time and mean delay but does increase the miles traveled by each vehicle. The increase in computational time is due to the larger number of



(a) Mean service rate

(b) Mean in-car travel delay $\delta - \omega$

(c) Mean waiting time ω

(d) Mean distance travelled

(e) Percentage of shared rides

(f) Mean computational time

Figure 5-6: Comparison of several performance metrics for varying number of sampled requests (No rebalancing, Reactive (0 samples), 200 samples, and 400 samples). The reactive method follows the algorithm of [5]. Each subplot corresponds to the vehicle capacity of 2 and 4 with the x-axis showing the fleet size (1000, 2000, and 3000 vehicles).

requests that enter the routing and assignment problem. Furthermore, since they are in the future, they can be combined with many different trips, which leads to

a potentially large number of feasible trips to be accounted for in the assignment. Nonetheless, the approach can be parallelized and would benefit from the large parallel servers available for fleet management companies.

To sum up, our experimental study confirms that the performance of a mobility-on-demand system with ride-sharing via our algorithm improves with knowledge of future demand. Yet, at a higher operational cost. For theoretical analysis of the approach, please refer to [5]

Chapter 6

Conclusion

This thesis showcases a reactive anytime optimal method with scalable real-time performance for assigning passenger requests to a fleet of vehicles of varying capacity. We quantify experimentally the trade-off between fleet size, capacity, waiting time, travel delay, and operational costs for low and medium capacity vehicles, such as taxis or vans in a large scale city dataset. Under the assumption of one person per ride, we show that 98% of the taxi rides currently served by over 13000 taxis could be served with just 3000 taxis of capacity four. We observe that vehicle capacity of two is sufficient for ride sharing when a small trip delay of two minutes is imposed. If a maximum delay of five minutes or more (comparable to the time spent retrieving a car from parking) is allowed, higher capacity vehicles 1) increase the service rate significantly, 2) reduce the waiting time, and 3) reduce the distance travelled by each vehicle. Our analysis shows that a ride-pooling service can provide a substantial improvement in urban transportation systems and that the system parameters such as vehicle capacity and fleet size depend on quality of service requirements and demand. We also show that we can utilize historical taxi trip data to anticipate future demand and pro-actively position vehicles to achieve a lower waiting time and travel delay.

Our algorithm is slightly limited because it does not fully utilize the vehicle fleet even when there are unassigned requests. This is due to our waiting time constraints which are needed to compute the which pairwise shareability graph. Despite this limitation, we are still able to achieve the high service rate discussed before. We

also show that we are able to rebalance the idle vehicles to reduce the effect of this limitation and improve the performance of the algorithm.

This work has far reaching implications for practical applications. Since less vehicles are needed to service most of the taxi demand due to optimal assignment and ride-sharing, taxi and mobility on demand companies can reduce their active vehicle fleet. This could reduce congestion and limit traffic on the road. Similarly, if there are less cars on the road, and these cars have a higher utilization, the amount of emissions per capita could also decrease. Reducing the size of a vehicle fleet for a MoD company can also cut their costs which may reduce the price of using a ride-sharing vehicle. This could drastically reshape the landscape of urban transit as ride-sharing and mobility on demand could become another modality of public transport. This would increase the connectivity of city and make regions where there may not be great public transit accessible due to the flexibility of automotive transport. Also, since the system we have developed is portable to cities other than New York, urban planners could use it to simulate how ride-sharing can affect the mobility of a neighbourhood. This can effect the urban landscape and give planners a better understanding how their city would get around. Likewise, if there was large scale adoption of ride-sharing enabled mobility on demand systems, limited congestion would cause urban planners to rethink how we utilize the space in our cities. We could less space dedicated to cars, and more space dedicated to people, to cafes, to restaurants. We could have dense urban centers with more plazas, more room to move, and more room to breath. We could have more room for businesses and parks. Enabling intelligent on-demand high-capacity ride-sharing can unlock a world of possibilities that would have an immediate impact on the way we commute, where we live, and the areas we visit.

6.1 Lessons Learned

From working on this project, I have learned a great deal about constrained optimization and techniques to transform a very large, seemingly irreducible problem

into something that can be more simply formulated and solved using well known, mathematically rigorous methods. On a more practical side, I learned a lot about dealing with a vast amount of data. Specifically, each year of historical taxi data included over 100 million trips (165 million in 2013, the year we used). I had to learn quickly that my normal pipeline for data processing would not be effective for such a large amount of data. A great deal of effort was spent figuring out how to process this data in a timely manner. Similarly on a practical side, I learned a lot about creating plots to convey information effectively. I am particularly fond of Fig. 5-2 since it was an interesting way to visualize the vehicular capacity all at once as a function of time. I am delighted to be a part of this project and hope to apply our ride-sharing method to multiple different problem domains.

6.2 Future work

In the future we would like to investigate a more sophisticated method for rebalancing idle vehicles that is able to react more quickly to anticipated changes in taxi demand. Currently our rebalancing method sends vehicles to areas where passengers went unserviced in the hope that demand in that area would increase again. Though we see great improvement in the all our metrics including, travel delay, waiting time, and serviced rate, we are interested in developing a more mathematically grounded approach with guarantees. We also would like to develop a new path finding algorithm that is able to optimally route vehicles through high demand regions whilst balancing congestion in crucial areas in the city. Our current method is stochastic and includes sampled future requests from our demand probability distribution to bias the path towards area of high demand, however we are interested in developing a less stochastic approach to that is able to utilize more knowledge about patterns in demand fluctuation to route the vehicles. Improving our prediction capabilities by utilizing novel machine learning techniques to enhance the responsiveness and robustness of our predictions is also being investigated as well as its implications in enhancing system efficiency.

We believe our approach is not only suited for assigning passengers to ride-sharing vehicles but also cargo to trucks. We are interested in investigating how our approach can be applied to logistics and package delivery. Package carriers usually carry multiple items with different pick up and drop off locations along with various time constraints. We believe that our algorithm could potentially improve the efficiency of package delivery in the future.

Appendix A: Tabulated Results

Vehicles	Capacity	Ω [s]	Δ [s]	Service Rate	Mean Waiting [s]	Mean In-Car Delay [s]	Mean Passengers	Shared Rate	Mean Travel [km]
1000	1	120	240	0.262978	81.5581	0	0.768697	0	182.027
1000	1	300	600	0.283784	234.689	0	0.835099	0	203.598
1000	1	420	840	0.287732	320.615	0	0.847364	0	207.773
1000	2	120	240	0.36998	76.3792	44.7869	1.16233	0.907206	185.032
1000	2	300	600	0.413232	219.236	134.524	1.50662	0.957515	196.476
1000	2	420	840	0.412036	325.386	171.178	1.5667	0.962714	196.663
1000	4	120	240	0.433255	73.5066	52.5421	1.37394	0.905983	185.513
1000	4 n.R.	300	600	0.373123	183.438	195.725	1.48346	0.942783	117.326
1000	4	300	600	0.580854	176.804	188.716	2.30771	0.943285	193.919
1000	4	420	840	0.603213	258.585	258.446	2.60321	0.950714	190.877
1000	10	120	240	0.440009	73.2917	52.804	1.39497	0.902917	185.683
1000	10	300	600	0.669158	157.273	197.282	2.68934	0.905023	194.029
1000	10	420	840	0.75182	203.156	287.461	3.37339	0.899895	189.824
2000	1	120	240	0.481458	77.7428	0	0.700887	0	154.727
2000	1	300	600	0.524316	220.625	0	0.769403	0	175.176
2000	1	420	840	0.532492	289.226	0	0.782802	0	179.546
2000	2	120	240	0.645492	73.3992	37.0945	1.00548	0.87967	155.225
2000	2	300	600	0.745878	201.348	121.656	1.33669	0.955198	164.91
2000	2	420	840	0.753914	288.845	153.138	1.40076	0.96388	165.58
2000	4	120	240	0.709123	71.3443	42.9121	1.11723	0.877091	154.697
2000	4 n.R.	300	600	0.625829	173.514	191.711	1.23814	0.940455	101.356
2000	4	300	600	0.911007	153.465	151.686	1.70709	0.925342	157.391
2000	4	420	840	0.937389	197.158	197.389	1.86932	0.931588	155.629
2000	10	120	240	0.713112	71.2705	43.1219	1.12366	0.875883	154.666
2000	10	300	600	0.944974	143.459	156.101	1.78505	0.899875	153.817
2000	10	420	840	0.977252	171.2	210.808	1.98942	0.895912	148.717
3000	1	120	240	0.652067	72.4532	0	0.63218	0	133.714
3000	1	300	600	0.734434	195.736	0	0.71695	0	156.49
3000	1	420	840	0.744333	251.324	0	0.727727	0	160.412
3000	2	120	240	0.794154	69.3053	28.0659	0.812734	0.798371	127.736
3000	2	300	600	0.929443	158.198	76.3427	1.02932	0.903125	139.298
3000	2	420	840	0.942151	191.746	87.6798	1.06026	0.913134	140.334
3000	4	120	240	0.829284	68.0709	33.162	0.857788	0.806376	124.355
3000	4 n.R.	300	600	0.797301	157.168	161.856	1.00192	0.92347	93.0818
3000	4	300	600	0.969812	136.944	112.643	1.14426	0.895421	119.019
3000	4	420	840	0.979149	162.45	137.158	1.20086	0.902623	117.996
3000	10	120	240	0.830674	68.0008	33.3181	0.859327	0.806087	124.147
3000	10	300	600	0.973502	133.96	127.549	1.17848	0.885291	112.411
3000	10	420	840	0.985835	154.013	164.524	1.26653	0.887157	110.053

Table 1: Average values of several performance metrics for ride sharing over a whole week with over 3 million requests. All experiments include rebalancing except for those indicated with (n.R.) next to the vehicle capacity. Ω is the maximum waiting time and Δ the maxim delay, including both in-car delay and waiting time. Service and shared rates are in per one.

Bibliography

[1] 2014 Yellow Taxi Trip Data. `https://data.cityofnewyork.us/view/gn7m-em8n`. Accessed: 2016-09-12.

[2] Empirical evaluation of a dynamic and distributed taxi-sharing system. *Proceedings of Conference on Intelligent Transportation Systems*, 2012.

[3] N.A.H. Agatz, A.L. Erera, M.W.P. Savelsbergh, and X. Wang. Dynamic ride-sharing: A simulation study in metro atlanta. *Transportation Research Part B: Methodological*, 2012.

[4] The Jin Ai and Voratas Kachitvichyanukul. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5):1693–1702, 2009.

[5] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. Ride-Vehicle Assignment and Analysis of the Benefits of High Capacity Vehicle Pooling. *Proceedings of the National Academy of Sciences*, 2017.

[6] Javier Alonso-Mora, Alex Wallar, and Daniela Rus. Dynamic Vehicle Routing with Sampled Future Requests for Autonomous Mobility on Demand. *Conference*, under review.

[7] Ph Augerat, Jose Manuel Belenguer, Enrique Benavent, A Corberán, D Naddef, and G Rinaldi. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. 1998.

[8] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

[9] Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European journal of operational research*, 202(1):8–15, 2010.

[10] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118, 2005.

[11] Carlos Carrion and David Levinson. Value of travel time reliability: A review of current evidence. *Transportation research part A: policy and practice*, 46(4):720–741, 2012.

[12] Han-wen Chang, Yu-chin Tai, and Jane Yung-jen Hsu. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5(1):3–18, 2009.

[13] J F Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 2006.

[14] Gonçalo Homem de Almeida Correia and Bart van Arem. Solving the user optimum privately owned automated vehicles assignment problem (uo-poavap): A model to explore the impacts of self-driving vehicles on urban mobility. *Transportation Research Part B: Methodological*, 87:64–88, 2016.

[15] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering Route Planning Algorithms. 2:117–139, 2009.

[16] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, 40(2):342–354, 1992.

[17] Jan Dethloff. Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *Or Spectrum*, 23(1):79–96, 2001.

[18] B Donovan and D B Work. Using coarse gps data to quantify city-scale transportation system resilience to extreme events. *arXiv.org*, 2015.

[19] Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *The European Journal of Operational Research*, 1990.

[20] Terry L Friesz, Javier Luque, Roger L Tobin, and Byung-wook Wie. TRAFFIC ASSIGNMENT DYNAMIC NETWORK CONSIDERED AS A TIME OPTIMAL CONTINUOUS CONTROL PROBLEM. 37(6):893–901, 2012.

[21] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.

[22] Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[23] E J Gonzales, C Yang, E F Morgul, and K Ozbay. Modeling Taxi Demand with GPS Data from Taxis and Transit. 2014.

[24] Dwight A Hennessy and David L Wiesenthal. Traffic congestion, driver stress, and driver aggression. *Aggressive behavior*, 25(6):409–423, 1999.

[25] M.E..T. Horn. Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C: Emerging Technologies*, 2002.

[26] Brian Kallehauge, Jesper Larsen, Oli BG Madsen, and Marius M Solomon. Vehicle routing problem with time windows. In *Column generation*, pages 67–98. Springer, 2005.

[27] S. Ma, Y. Zheng, and O. Wolfson. T-share: A large scale dynamic taxi ridesharing service. *Proceedings of IEEE ICDE*, 2013.

[28] Hani S Mahmassani. Dynamic network traffic assignment and simulation methodology for advanced system management applications. *Networks and spatial economics*, 1(3-4):267–292, 2001.

[29] D K Merchant and G L Nemhauser. Optimality conditions for a dynamic traffic assignment model. *Transportation Science*, 12(3):183–199, 1978.

[30] Snezana Mitrovic-Minic. *The dynamic pickup and delivery problem with time windows.* Simon Fraser University, 2001.

[31] Fermín Alfredo Tang Montané and Roberto Diéguez Galvao. A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers & Operations Research*, 33(3):595–619, 2006.

[32] L Moreira-Matias, J Gama, and M Ferreira. On predicting the taxi-passenger demand: A real-time approach. *International Conference on Artificial Intelligence*, 2013.

[33] Pallavi Pant and Roy M Harrison. Estimation of the contribution of road traffic emissions to particulate matter concentrations from field measurements: a review. *Atmospheric Environment*, 77:78–97, 2013.

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

[35] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11, 2013.

[36] Samitha Samaranayake, Walid Krichene, Jack Reilly, Maria Laura Delle Monache, Jean-Baptiste Lespiau, Paola Gaotin, and Alexandre Bayen. System Optimal Dynamic Traffic Assignment with Partial Compliance (SO-DTA-PC). In *Proceedings of the American Control Conference (ACC)*, 2015.

[37] P. Santi, G. Resta, M. Szell, S. Sobolvesky, S.H. Strogatz, and C. Ratti. Quantifying the benefits of vehicle pooling with shareability networks. *PNAS*, 2014.

[38] David Schrank, Bill Eisele, and Tim Lomax. Texas Transportation Institute 2012 urban mobility report. *College Station, TX: Texas Transportation Institute, A&M University*, 2012.

[39] K. Spieser, S. Samaranayake, W. Gruel, and E. Frazzoli. Shared-vehicle mobility-on-demand systems: a fleet operator's guide to rebalancing empty vehicles. *Transportation Research Board 95th Annual Meeting, Washington, D.C.*, 2016.

[40] K. Spieser, K. Treleaven, R. Zhang, E. Frazzoli, D. Morton, and M. Pavone. *Toward a systematic approach to the design and evaluation of automated mobility-on- demand systems: a case study in Singapore.* Road Vehicle Automation, 2014.

[41] Andreas Stenger, Daniele Vigo, Steffen Enz, and Michael Schwind. An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science*, 47(1):64–80, 2013.

[42] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*, volume 18. Siam, 2014.

[43] Marco Veloso, Santi Phithakkitnukoon, and Carlos Bento. Sensing urban mobility with taxi flow. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pages 41–44. ACM, 2011.

[44] R. Zhang and M. Pavone. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. *Proceedings of Robotics: Science and Systems Conference*, July 2014.

[45] Athanasios K Ziliaskopoulos. A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation science*, 34(1):37–49, 2000.

[46] Athanasios K Ziliaskopoulos. Foundations of Dynamic Traffic Assignment : The Past , the Present and the Future. *Mathematical Programming*, 1(3):233–265, 2001.