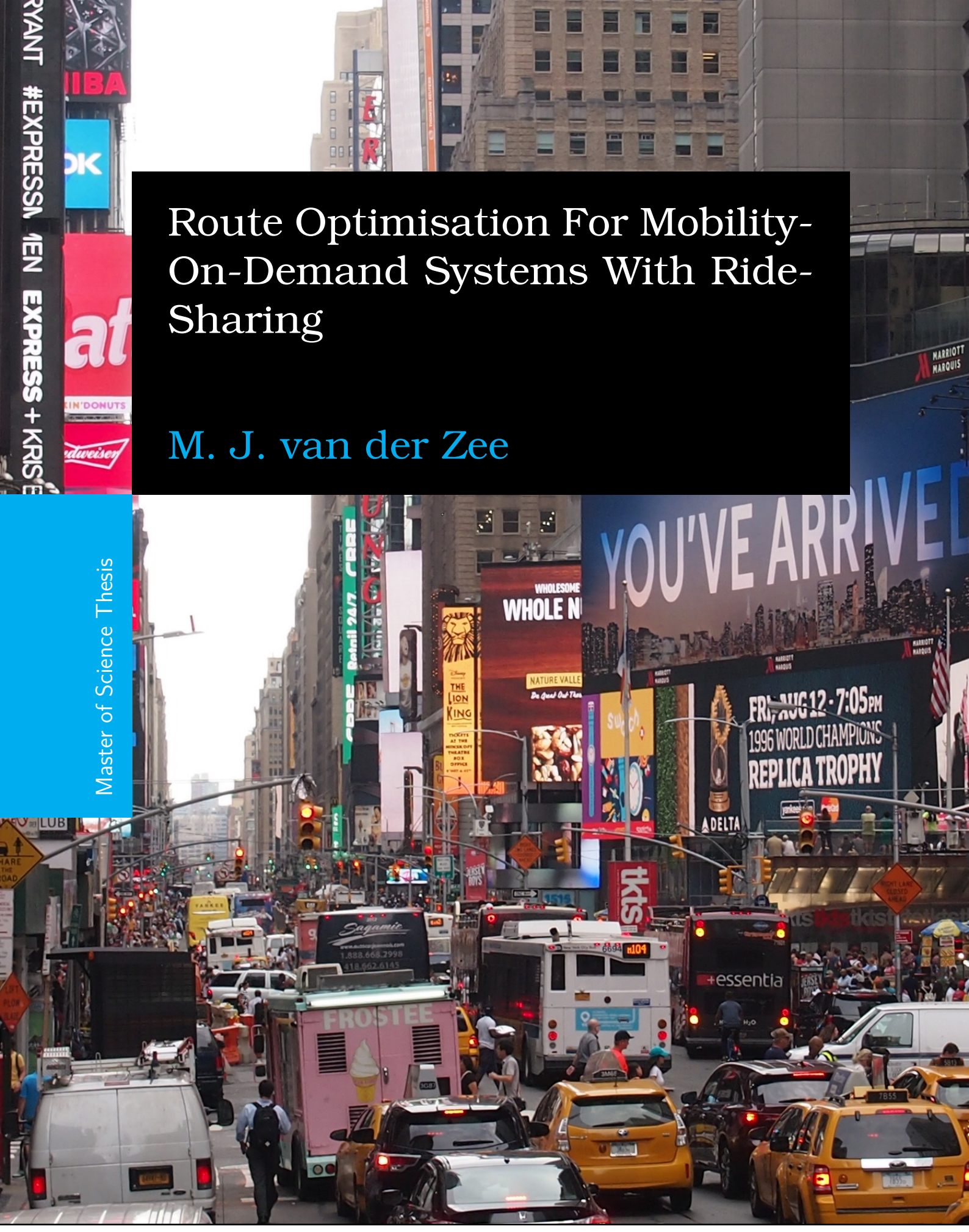
Route Optimisation For Mobility-On-Demand Systems With Ride-Sharing

M. J. van der Zee

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Route Optimisation For Mobility-On-Demand Systems With Ride-Sharing

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

M. J. van der Zee

June 19, 2018

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

This work has been conducted in cooperation with the National University of Singapore (NUS) and the Singapore-MIT Alliance for Research and Technology (SMART) under the Future of Urban Mobility (FM).

Abstract

Privately owned cars are an unsustainable mode of transportation, especially in cities. New Mobility-on-Demand (MoD) services should offer a convenient and sustainable alternative to privately owned cars. Notable in this field is the recent uprise of ride-sharing services such as offered by companies like Uber and Grab. Such services, especially when allowing for multiple passengers to share a vehicle, could potentially be a valuable addition to existing modes of public transport to offer fast and sustainable door-to-door transportation.

The optimisation of vehicle routes for a MoD fleet is a complex task, especially when allowing for multiple passengers to share a vehicle. Recent studies have presented algorithms that can optimise routes in real-time for large scale ride-sharing systems, but have left opportunities to further enhance fleet performance. The redistribution of idle vehicles towards areas of high demand and the utilisation of high capacity vehicles in a heterogeneous fleet has received little attention. This work presents a method to continuously redistribute idle vehicles towards areas of expected demand and an analysis of fleets with both buses and regular vehicles. Furthermore, a method is proposed to optimise vehicle routes while taking into account vehicle capacities and the future locations of vehicles in anticipation to predicted demand.

In simulations with historical taxi data of Manhattan, 99.8% of transportation requests can be served with a fleet of 3000 vehicles with an average waiting time of 57.4 seconds, and an average in-car delay of 13.7 seconds. Compared to earlier work with a fleet of 3000 vehicles, a decrease in ignored requests of 95% is obtained, with a 86% decrease in average in-car delay and a 37% decrease in average waiting time. For a small fleet of 1000 small buses of capacity 8 still 84.6% of requests can be served with an average waiting time of 141 seconds and an average in-car delay of 269 seconds. In comparison to prior work, a decrease in ignored requests of 15% is obtained, with a 14% decrease in average in-car delay and a 2% decrease in average waiting time. A heterogeneous fleet of 1000 vehicles consisting of 500 buses and 500 regular vehicles using this new approach can serve approximately the same number of passengers as a homogeneous fleet of 1000 buses using earlier presented algorithms.

Table of Contents

List of Figures

List of Tables

Preface

Most of us experience the struggles of commuting. The often uncomfortable time we spend every day to travel back and forward between our work. Time which we would much rather like to spend in other ways. My motivation to improve the way we travel around lies in exactly this experience. This document is the graduation thesis for my Master of Science at the Delft University of Technology (TU Delft) and is one of the results of a year long endeavour to make urban mobility a bit more efficient and comfortable.

From August 2017 to June 2018 I have researched and implemented ideas to make urban transportation more efficient at the Singapore-MIT Alliance for Research and Technology (SMART) in Singapore. Why Singapore? Because Singapore is a country where transportation initiatives thrive. A place where autonomous vehicles are allowed to drive among the public and a place where it is very affordable to take rides with Grab drivers to get first hand experience of the strengths and weaknesses of these platforms.

This document is merely a report of the results of my year long research, and covers just the tip of the iceberg of the journey that I went through to complete this document. A journey that has been though at times and included many failed experiments, crashed computers, and relentless days of debugging. I hope it offers you, the reader, some interesting insights and a contribution to your knowledge in the possibilities of improving the efficiency of urban mobility.

Acknowledgements

I would like to thank my supervisor Javier Alonso-Mora for his guidance during the writing of this thesis and for the introduction to SMART.

I would also like to thank the people at the SMART Future of Urban Mobility lab for kindly welcoming me to the team. More specifically I would like to thank Marcelo H. Ang Jr. for giving me the opportunity to join SMART, Malika Meghjani for guiding me through the start of this project, Alex Wallar for the enjoyable and fruitful cooperation, and Shashwat Verma for being a great friend both in and outside of the lab.

Finally I would like to thank my parents for giving me the opportunity to pursue this study, and Charlotte. I cannot emphasise enough my gratitude for their love, their pride in even my smallest and insignificant achievements, and their support of my dreams, even if that takes me halfway around the world.

Delft, University of Technology
June 19, 2018

M. J. van der Zee

"Enthusiasm is the yeast that makes your hopes shine to the stars. Enthusiasm is the sparkle in your eyes, the swing in your gait. The grip of your hand, the irresistible surge of will and energy to execute your ideas."

— *Henry Ford*

Chapter 1

Introduction

In modern society, transportation is responsible for a significant part of the total energy consumption. In 2016, transportation in the USA was responsible for roughly 29% of the total USA energy consumption. In that same year, 92% of the energy used by the USA transportation sector was provided by non-renewable petroleum products [3]. The transition to sustainable methods of transportation pose a serious challenge. Urban centres depend on transportation systems for their economic prosperity and liveability. This comes at a cost: transportation leads to air pollution, green house gas emissions, congestion and road accidents. It is estimated that air pollution is responsible for 800,000 deaths annually in urban centres [4]. Also, urban transport accounts for 40% of the total emission of CO_2 [5]. Furthermore, road accidents are responsible for 1.25 million deaths annually and are the primary death cause among people aged between 15 and 29 years old [6].

The primary cause of these issues is the use of private motor vehicles [4]. The number of motor vehicles in the world is increasing explosively. The reason is a combination of inadequate public transportation and increasing accessibility of vehicles due to dropping price and increasing income in large parts of the world. The major advantage of personal cars over other modes of transportation is that it offers comfortable, efficient, anytime door-to-door transportation. For this reason, many people who can afford it choose to buy a car. In the USA, approximately 0.8 cars are registered for every inhabitant [7]. Approximately 20% of all cars are registered in the USA, while the USA only comprises 5% of the world population [8]. If the rest of the world would follow this trend, this would most likely make the detrimental effects of motor vehicles much worse. Indications are that this is already happening. The six largest cities in India for example saw a doubling in their population between 1981 and 2001, yet in the same period, the number of motor vehicles increased by a factor 8 [4].

1-1 Mobility-on-Demand

Recent developments in Mobility-on-Demand (MoD) such as ride-sharing and car-pooling have focussed on services offering the convenience of a privately owned car without the associated detrimental effects. Transportation by road vehicles could be a valuable part of urban

transportation infrastructure, filling the gaps where public transportation is not time efficient. It can either be used as a first/last mile alternative, complementing existing modes of public transportation, or as a dedicated door-to-door transportation mode. The development of autonomous vehicles further amplifies the interest in such services.

Perhaps most notable in the field of MoD is the recent uprise of ride-sharing services. Ride-sharing companies such as Uber, Lyft, DiDi and Grab have seen significant growth in recent years. Ride-sharing companies have transformed the taxi industry by the introduction of e-hailing. These companies promise to alleviate congestions by taking cars off the road by dissuading the use of privately owned cars. Yet it seems that rather the contrary is true. Ride-sharing companies lead to more vehicles on the road and actually slow down traffic in many places [9, 10]. In the major cities in the USA, 49% to 61% of trips made with ride-sharing would not have been made at all, or by bike, on foot or by public transportation. This has lead to a drop in bus and rail usage of 6% and 3% respectively [10]. It furthermore causes an increase in vehicles on the road. Since the uprise of ride-sharing, an increase of 59% of vehicles and an increase of 81% of vehicles without passengers was reported in the central business district of Manhattan [9].

It seems therefore that ride-sharing is an ineffective and unsustainable alternative to privately owned cars. It is also apparent however that the vehicles in ride-sharing fleets are inefficiently used, since [9] reports that many vehicles are driving around empty. Ride-sharing companies are trying to increase fleet efficiency by combining multiple passenger trips in a single vehicle, and indicating to drivers where demand is high. A significant challenge remains however in computing vehicle routes to efficiently serve demand, and to allow multiple passengers to share vehicles.

1-2 Research Objective

The objective of this work is to develop an enhanced algorithm to dynamically route vehicles in a large scale MoD system in which multiple passengers can share the same vehicle. The focus is on improving the fleet performance in comparison to prior work in an urban environment with high densities of transportation requests and a fleet with a limited number of vehicles. The primary performance indicator is the number of passengers that can be served by the fleet, subject to a set of service constraints. To achieve this, the focus in this work more specifically is on:

- Continuous redistribution of idle vehicles towards areas of expected demand.
- Utilisation of both regular vehicles and high capacity buses in a heterogeneous fleet.
- Optimisation of vehicle routes which takes into account vehicle capacities and anticipates future demand.

1-3 Contributions

The contributions presented in this work expand on earlier published algorithms for real-time dynamic route optimisation for a MoD fleet in which vehicles can be shared among multiple

passengers. For this work, the state-of-the-art was reimplemented and enhanced with several features to improve fleet performance. Specifically, this work presents:

- A method to continuously optimally redistribute idle vehicles over an operating area in anticipation to estimated future demand.
- A study into the effect of combining high capacity buses and regular vehicles in a single fleet.
- A method to assign candidate vehicle routes while taking into account estimated future demand and vehicle capacities.

1-4 Chapter Overview

In Chapter 2, an overview is given of the existing literature. Chapter 3 formally introduces the problem in mathematical terms and gives an overview of the implemented algorithms that are used throughout this thesis. In Chapter 4 a method is presented to effectively distribute idle vehicles over an operating region according to estimated demand. In Chapter 5 the effect of different vehicle capacities in the fleet is studied in both heterogeneous and homogeneous fleets. In Chapter 6 an improved method is presented to assign schedules to vehicles while taking into account vehicle positioning in anticipation to future predicted requests and vehicle capacities in a heterogeneous fleet. Finally in chapter 7 conclusions and recommendations for future work are presented.

Chapter 2

Related Works

The problem to optimally route a fleet of vehicles is far from new and a rich variety of literature is available on solutions to such problems.

2-1 Vehicle Routing Problem

The problem studied in this work is related to so called vehicle routing problems (VRPs), which are concerned with finding optimal vehicle routes. The VRP is a widely studied problem that was first introduced in [11] in 1959 as the truck dispatching problem. The problem is concerned with finding the optimal route for a fleet of vehicles to visit a set of customers from a depot. In this case it is presented as a set of gasoline delivery trucks required to visit a set of gas stations. Each gas station has a specified demand, and the vehicles have a limited capacity. In a VRP, vehicles have to visit a number of nodes, and the problem is to find a feasible optimum order in which vehicles visit these nodes subject to some cost function. Such a cost function could be distance travelled or time until completion. A solution to a VRP is illustrated in Figure 2-1.

The VRP is a combinatorial optimisation problem and can be formulated as an integer linear programming (ILP). Many different ILPs formulations have been proposed for the VRP. The VRP is defined for m heterogeneous vehicles with capacity Q and n customer nodes with an associate demand q_i on a complete undirected graph $G = (V, E)$. The graph contains the set of vertices $V = \{0, \ldots, n\}$ and the set of edges $E = \{(i, j) : i, j \in V, i \neq j\}$. Vertex 0 represents the depot node at which all routes of the vehicles start. The edges E represent the routes between the nodes. Every edge has an associated cost, which is denoted as $c_{i,j}$. This cost can for example be the travel time, or route distance. Notice that in this case, the cost matrix is symmetric. This means that the route between nodes i and j has the same cost regardless of direction ($c_{i,j} = c_{j,i}$). Would this not be the case, then the problem is defined on a directed graph $G = (V, A)$. The problem is now to find routes for all m vehicles that satisfy the following constraints:

1. All routes start and finish at the depot (vertex 0).
2. Each customer node is visited once by one vehicle.
3. The vehicle capacities are not exceeded.
4. The total cost of the routes is minimised.

VRPs are NP-hard [1] and are notoriously hard to solve. Exact solutions can generally only be found for problem instances with limited numbers of vehicles and customers. A lot of the proposed methods for solving VRPs therefore rely on heuristics. An overview of both exact formulations and formulations based on heuristics and their performance is given in [1] and [12]. A review of this problem where requests dynamically enter the system is given in [13].

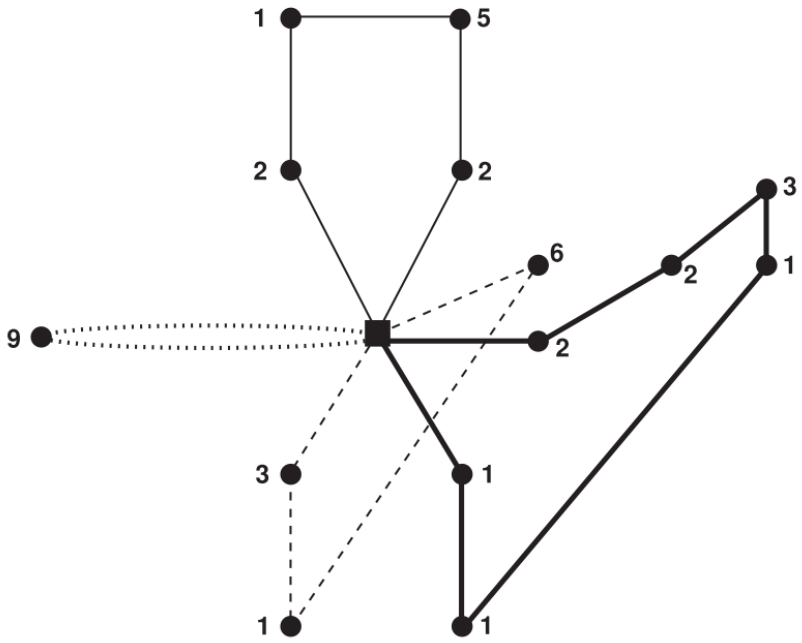


Figure 2-1: A solution to the VRP with 14 customer nodes and 4 vehicles with capacity 10. The demand at each customer is shown next to the customer nodes. The different vehicle routes are represented by different line styles. Taken from [1].

2-2 Dial-a-ride Problem

The problem that is subject of study in this thesis is a special case of the VRP. Instead of distributing agents from a single depot to many different locations, it involves no depot and agents that are both picked-up and drop-off. This is a special kind of VRP referred to as the vehicle routing problem with pick-up and delivery (VRPPD). Another important difference is that people are transported instead of freight. This requires strict constraints on pick-up and drop-off time. This special subclass of VRPPDs is often referred to as the dial-a-ride problem (DARP). The vehicle capacities are constrained, but multiple passengers can be

in a vehicle at the same time. An example is illustrated in Figure 2-2. Additionally, the problem is dynamic meaning that not all transportation requests are known beforehand but dynamically enter the system. Examples of formulations are presented in [14], [15], [16] and [17]. An overview of the dynamic DARP is presented in [18].

The problem instances that can be solved using the formulations proposed for the DARP are limited. The problem instances that can be solved to optimality using exact formulations are in the order of 50 requests. Even when relying heavily on heuristics it takes a matter of minutes to solve problem instances of roughly 100 requests and 10 vehicles [17].

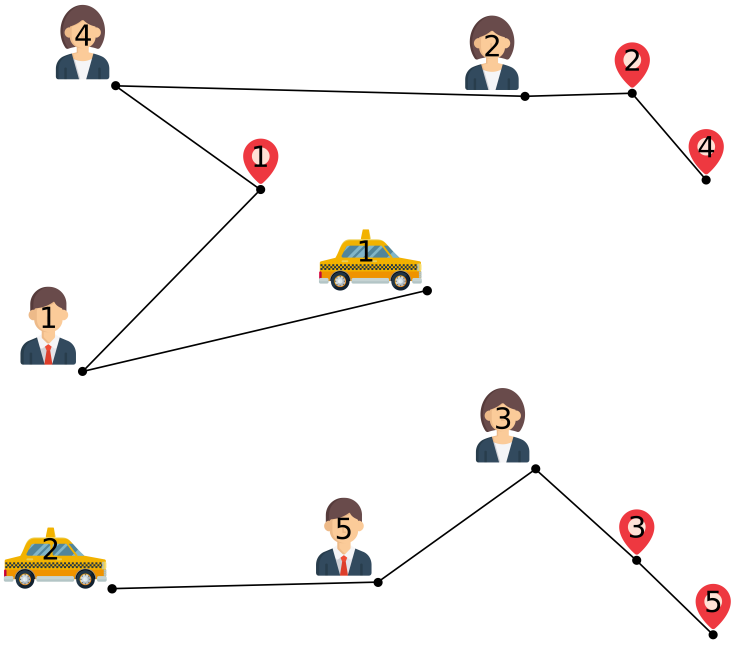


Figure 2-2: An example of a solution to the DARP with a fleet of 2 vehicles and 5 transportation requests. The humans represent the request origins and the red markers represent their respective destinations.

2-3 Large Scale Ride-Sharing

In cities, the problem sizes that can be solved using the proposed exact or heuristic approaches to the VRP are not sufficient. The number of vehicles in a fleet and number of transportation requests are presumably much larger. Several publications have looked into formulations that can handle such large problem instances.

One of the first practical attempts of a routing algorithm for large scale taxi-sharing is presented in [19] and later elaborated in [20]. The algorithm proposed is focused on a rush hour

situation when the number of vacant vehicles is low. When a request is placed, the proposed algorithm searches among vehicles that are already serving another request, taking into account the destination of the passenger. It finds the vehicle that can serve the request with the minimum added travel distance to the request that it is already serving. It does this for every new request in order of the time that the request was placed (a greedy strategy). Although this is a greatly simplified approach, it reports that in a simulation based on real taxi data, 3 times as many requests could be served and the total distance travelled by the vehicles was decreased by 11%.

A greatly improved algorithm for ride-sharing was proposed by [21]. This paper presents a case study of Manhattan which shows that a large fraction of rides could potentially be shared among customers with minimum time delay. The notion of shareability is introduced. Trips are said to be shareable if there exists a route connecting the origins and destinations of the individual requests with a delay no more than a preset value. In the paper, the maximum number of requests to be combined in a single vehicle is limited to 2 to limit the computational complexity. It is stated that the problem can be solved heuristically for a maximum number of requests to be combined of 3, and that the problem is intractable for 4 or more. With a limit of 2 shared trips, the problem can be solved exactly. The algorithm searches for all possible pairs of requests that can be combined under the maximum delay constraint to construct a shareability network. Using this shareability network, the actual trips are then chosen to optimise for either maximum number of shared trips, or minimum total travel time. To do this, the algorithm computes either a maximum matching or weighted maximum matching on the shareability network. This work furthermore uses historical taxi data and introduces an Oracle model in which all past and future requests are known which serves as an upper bound of the possible performance of such a Mobility-on-Demand (MoD) system. Using this Oracle model, the sharing of a maximum of 2 requests, and a maximum allowed delay time of 1 minute, this paper claims that 94.5% of all trips are shareable. In a dynamic application (without the Oracle model), using a horizon of 1 minute, this value drops to less than 30%. Using a maximum allowed delay of 5 minutes however, the total vehicle travel time is decreased by 32%.

The advantage of sharing vehicles in a MoD service was further emphasised by [2]. This work presents a greatly enhanced algorithm compared to earlier work that handles high capacity vehicles (up to 10 passengers) in real-time and rebalances idle vehicles to high demand areas. Furthermore it handles large numbers of requests and vehicles. Like in the work of [20], the proposed algorithm is capable of rerouting vehicles that are already en-route whenever new requests come in. The algorithm works in four steps of which part is based on the work by [21]. Firstly it is determined which requests can be pairwise combined while satisfying a maximum delay constraint, and which vehicle can serve which request individually based on a maximum waiting time constraint and a vehicle capacity constraint. Secondly, this information is used to explore which larger groups of requests can be combined in a vehicle while still meeting the delay, waiting time and vehicle capacity constraints. Thirdly, an ILP is performed to find the optimal vehicle routes among the found feasible routes. Here, the cost function is defined as the sum of all delays with a large penalty for unassigned requests. Finally, possible remaining idle vehicles are sent to areas of high demand. In this case this is tractable for large numbers of vehicles and requests (both in the thousands). This is because the problem is decoupled into first checking for feasible trips, and subsequently solving the resulting ILP of reduced dimensionality. This algorithm was tested in a case study on Manhattan, New York using

real taxi data and showed that 3000 vehicles (current fleet size is 13000) with a capacity of 4 could serve 98% of all taxi rides in Manhattan with a mean waiting time of 2.7 minutes and a mean delay of 2.3 minutes.

A reimplementation of the work presented in [2] is used as the basis for further studies in this work. A more elaborate overview of the method is presented in Chapter 3. The above mentioned works only take into account requests currently placed in the vehicle route optimisation, but do not take into account how these routes affect vehicle positioning for serving future requests. A new method to optimise vehicle routes, which takes into account requests currently in the system while also anticipating future requests is presented in Chapter 6.

2-4 Heterogeneous Fleet

The work discussed so far has focussed on route optimisation for a fleet with only one type of vehicle. The problem which involves vehicles with varying route costs and capacities is known as the heterogenous fleet VRP. Several methods have been proposed in literature to generalise the VRP so that it allows for a heterogeneous fleet. An early overview and comparison of methods is given by [22]. All algorithms presented in this work are based on heuristics though, as it was found at the time that even for moderate problem sizes it is too difficult to solve exactly. A more recent overview of the approaches to solve the heterogenous fleet VRP is given by [23], and alternative formulations are presented in [24] and [25]. In [17] a compact ILP for the DARP is formulated with multiple depots and a heterogeneous fleet that is solved to optimality in a matter of seconds using an exact branch and cut (BC) algorithm. The problem instances that can be solved using these strategies however are too small for a realistic MoD fleet. The large scale MoD algorithm presented in [2] handles vehicles of different capacities, and shows results for homogeneous fleets of varying capacities. It does not present results however for heterogeneous fleets, or a method to optimise vehicle routes that takes vehicle capacities into account. An analysis of heterogeneous fleets for a large scale MoD system is presented in this work in Chapter 5. A strategy to assign vehicle routes based on anticipated future requests which takes into account vehicle capacities is presented in Chapter 6.

2-5 Rebalancing

Rebalancing is the redistribution of idle vehicles from areas of low demand towards areas of high demand. This is required since vehicles tend to build up in areas of low demand, and are depleted in areas of high demand. Rebalancing should lead to a better match between demand and supply of vehicles. The redistribution of idle vehicles has been covered in several other works. The majority of this work has focussed however on rebalancing vehicles in one-way car sharing schemes such as car2go and Zipcar. Such systems experience similar mismatches between vehicle supply and demand. Many of these works however focus on infrequent rebalancing (in the order of several times a day) [26], the practical implications of the use of human operators to rebalance vehicles [27], or on theoretical formulation and experimentation of the optimisation problem [28]. Prior work has also studied how vehicles

can be redistributed to fixed stations by drivers employed by a fleet manager [29, 30]. The most important difference is that vehicles in such systems are parked after usage by the users, and require a human operator to physically move to the vehicle to perform rebalancing. The applicability of these prior studies to a MoD service with continuous route optimisation as studied in this work is therefore limited.

Other studies have focussed on rebalancing in a MoD system, in which vehicles can redistribute themselves without intervention of a human operator. Much of this research shows that using such a strategy leads to a significantly reduced required fleet size to serve a fixed demand with a similar quality of service [31, 32]. However, these works often use long rebalancing intervals [31] or use simplified models to simulate demand and vehicle movement [32].

Rebalancing strategies for the same MoD system as discussed in this work are presented in [2] and in [33]. In both works, idle vehicles are continuously rebalanced. In [2] a simple strategy is used which rebalances idle vehicles to areas with ignored requests. If idle vehicles are available, an idle vehicle is dispatched towards the location of every request that could not be picked-up within a specified maximum waiting time. A more advanced rebalancing strategy is presented in [33]. This approach dispatches idle vehicles towards predicted demand learned from historical taxi data. This strategy however requires elaborate historical data on taxi rides to be available. Furthermore in both works, a relatively large number of ignored requests remain even when idle vehicles are available in the operating area. An enhanced rebalancing strategy is presented in Chapter 4. This method does not require historical taxi data, and yields a significant decrease in ignored requests, average passenger waiting time and delay over the methods presented in [2] and [33].

Chapter 3

Preliminaries

At the time of writing, the algorithm presented in [2] yields the best results and is the closest to a practical implementation for a real time ride-sharing application. To conduct experiments, the work of [2] was reimplemented. This was primarily implemented in Go, which was chosen for its combination of performance and brevity. Parts of the codebase are implemented in C++ in combination with the Gurobi optimisation library to solve the integer linear programmings (ILPs) (since no Gurobi interface is available for Go). In this chapter, the problem is formulated more formally in mathematical terms, and an overview is given of algorithms used in further experimentation.

3-1 Problem Formulation

A set of transportation requests $\mathcal{R} = \{r_1, \ldots, r_n\}$ is considered. A transportation request is defined as a person wanting to travel from an origin to a destination. In this work, requests are considered that are placed in real-time. A request r is defined by a tuple $\{o_r, d_r, t_r^r\}$, in which o_r is the request origin, d_r is the request destination and t_r^r is the time that the request was placed.

A fleet of vehicles $\mathcal{V} = \{v_1, \ldots, v_m\}$ is considered. Each vehicle v has an associated capacity κ_v and a state defined by the tuple $\{q_v, \mathcal{P}_v\}$. Here, q_v is the location of the vehicle, and $\mathcal{P}_v$ is a set of passengers that it has on board. A passenger is a request that has been picked up by a vehicle.

Furthermore, a schedule S is defined. All feasible schedules are collected in the set $\mathcal{S}$. Vehicles move according to their assigned schedules. A schedule is defined by a sequence of request pick-up and drop-off events. This is an abstraction level above the actual route that a vehicle is following, and serves as a set of waypoints that are visited by the vehicle in order.

Also, a trip $T \subseteq \mathcal{R}$ is defined as a set of requests. All feasible trips are collected in the set $\mathcal{T}$. This is an abstraction level higher than a schedule, and defines requests to be served in a combined schedule by a single vehicle. There may be many possible schedules that can be

assigned to a vehicle to serve a specific trip. Also, a trip may have more than one candidate vehicle by which it can be served.

An operating area is considered which comprises a road network. This is the region within which requests are considered, and vehicles operate. Vehicles travel only within the boundaries of this area and all request origins and destinations are in the operating area. The travel time between two locations q_i and q_j in the operating area is denoted by $\tau(q_i, q_j)$.

Furthermore a waiting time ω_r and a delay δ_r are defined for all requests. The waiting time for a request r is defined as $\delta_r = t_r^p - t_r^r$, in which t_r^p is the pick-up time. The delay is defined as $\delta_r = t_r^d - t_r^r - \tau(o_r, d_r)$, in which t_r^d is the request drop-off time. This is the difference between the theoretical earliest arrival time at the request destination and the actual drop-off time. The theoretical earliest arrival time at destination occurs when there is a vehicle available exactly at the location of a request at the time that the request is placed, and this vehicle transports the request directly from the origin to its destination.

Requests are placed continuously. Schedules are assigned to vehicles in ψ intervals. At the time of an assignment, all requests in the system are considered that have not yet been picked up and have not yet been ignored. An ignored request, or walk-away, takes places when a request has to wait longer than a predefined time before a vehicle can come to pick it up. Because vehicles can be assigned new schedules every assignment time, the vehicle that comes to pick up a specific request can change until it is actually picked up.

Three different vehicle statuses are defined. A vehicle can remain idle, it can be assigned a schedule to pick-up and drop-off passengers, or it can be assigned to rebalance. A rebalancing vehicle is assigned a location in the operating area to which it will travel without serving requests.

The problem is to find an assignment of schedules to vehicles, and vehicles to rebalancing locations so that the vehicle fleet serves the requests in an optimal way while respecting a possible set of service constraints. That is, the vehicle assignment minimises a defined cost function.

3-2 Simulation

To asses the performance of a fleet using new routing algorithms, the fleet must be simulated under realistic circumstances. Firstly, demand is simulated over an operating area. Simply spawning requests uniformly randomly over some road map would not suffice for this purpose, since this does not capture strong fluctuations in demand such as encountered in the real world during for example rush hour. Secondly, the motion of the vehicles over a road map is simulated. For this purpose a simulator was implemented. A visualisation of the simulator can be found in Figure 3-1.

3-2-1 Demand

For the simulation of demand, a real Manhattan taxi data set is used, which stores the coordinates of origins and destinations and the pick-up time of taxi trips in Manhattan for several years [34]. Since the dataset only contains the actual pick-up time, request placement

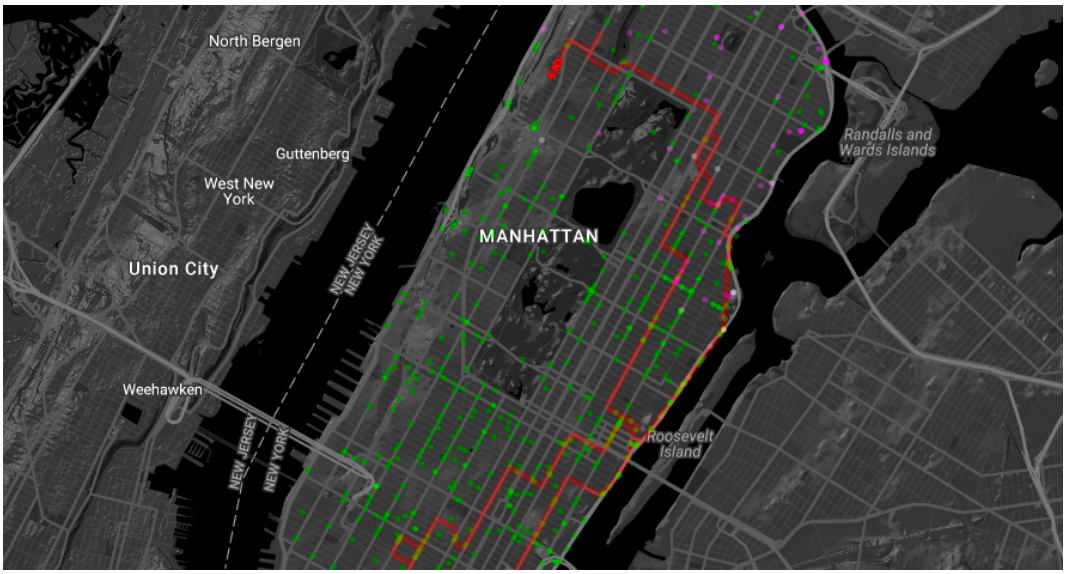


Figure 3-1: A snapshot of the simulator. Green dots represent vehicles that either have passengers on board or are on their way to pick up passengers. Pink dots represent vehicles that are driving towards areas of expected demand with no passengers on board. Grey dots represent vehicles that are idle. The snapshot furthermore shows the location and history of one of the vehicles in the fleet (represented by the red car).

time is simulated at the time of the pick up in the dataset. Requests are spawned at the pick-up time and origin stored in the dataset. It is furthermore assumed that every trip contains only 1 passenger.

3-2-2 Vehicle Movement

The Manhattan taxi data set contains no information on the route that is taken by the vehicle. It does however contain information on the travel time between the request origin and destination. This data allows to estimate travel times between nodes in the city. A graph of Manhattan with estimated travel times is provided by the authors of [21]. This graph contains 4091 nodes at intersections between roads and 9452 edges between connected nodes. Every edge has an associated stored average travel times for every hour of the day for weekdays, and Saturdays and Sundays. The path of the vehicles is found by performing a shortest path search on this graph.

3-2-3 Vehicle Path And Travel Time

For the optimisation, it is necessary to compute the shortest paths and associated estimated travel times between different locations in the city. To obtain these travel times, one option would be to store the road network offline and to store an average travel time over different sections in to road network. When the travel time between two points is queried, a search algorithm could then search for the shortest path between the points. Modern computers should be able to compute this in a relatively short time. A problem with this method however is that route optimisation requires the computation of paths and travel times between a lot

of points. This is partially solved by memoisation, since a lot of travel times are repeatedly queried for different schedules. Still however this makes the algorithm very slow. To overcome this problem, all the travel times are precomputed. For this, the graph provided by [21] is used. From this graph, a new, complete graph is created offline that stores the shortest paths and associated travel times between all 4091^2 pairs of nodes at every hour of weekdays and Saturdays and Sundays. This data can subsequently be queried during online route optimisation.

3-3 Base Method

As discussed in Chapter 2, the most promising algorithm for large scale ride-sharing is presented in [2]. The algorithm makes the problem tractable by splitting up the finding of schedules in several decoupled problems. Given enough time, this method returns guaranteed optimal vehicle routes. This is done by defining service constraints in the form of a maximum waiting time Ω and a maximum delay Δ, so that $\omega_r \leq \Omega \quad \forall r \in \mathcal{R}$ and $\delta_r \leq \Delta \quad \forall r \in \mathcal{R}$. Any schedules that cannot meet these constraints are not further considered and this allows in steps to effectively prune possible schedules. The finding of schedules is then split up in the following steps:

1. Construction of the Request-Vehicle (RV) graph: check which vehicles can serve which requests individually while respecting the maximum waiting time, and which pairs of requests could potentially be combined in a trip.
2. Construction of the Request-Trip-Vehicle (RTV) graph: from the RV graph incrementally collect requests in larger trips and check if feasible schedules exists by which these trips can be served.
3. Assignment of vehicles to trips: using the RTV graph, find the optimal assignment of trips to vehicles so that a certain cost function is minimised.
4. Redistribute idle vehicles: Optimally assign remaining idle vehicle to rebalancing locations so that a separate rebalancing cost function is minimised.

These steps are also visualised in Figure 3-2. The graphs created in these steps store trips. For every linked trip-vehicle pair, a specific schedule is stored that is optimised against a specific cost function. These steps will be further elaborated in the subsequent section.

3-3-1 Schedule Computation

The most important and computationally expensive step in the algorithm is to compute feasible and optimal schedules for a trip-vehicle pair. For every trip-vehicle pair there are often many possible different schedules by which that vehicle can serve the trip. The number of possible schedules increases when the trip size increases. As an example, for a trip $\{r_1, r_2\}$ two equally valid candidate schedules are (o_1, d_1, o_2, d_2) and (o_1, o_2, d_1, d_2). There are however several requirements that every schedule should meet:

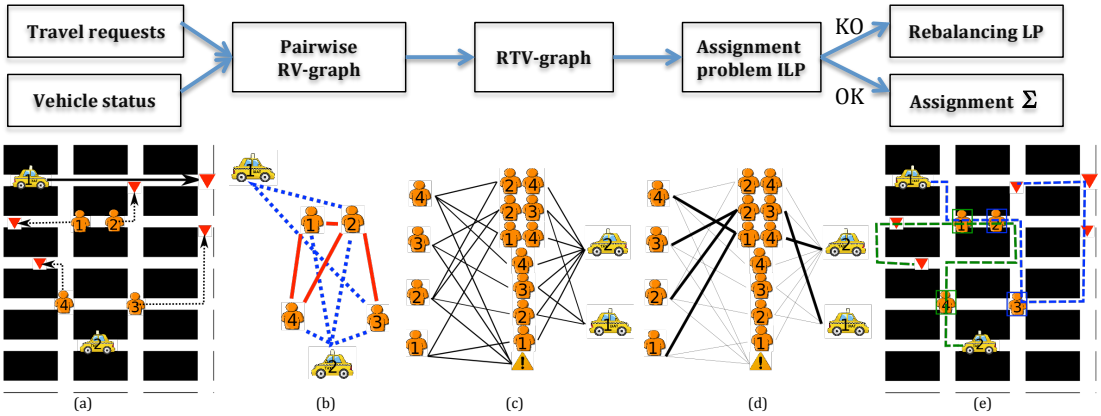


Figure 3-2: Schematic overview of the method used for the assignment of vehicles to requests. (a) Example of a street network with two requests (orange human = origin, red triangle = destination), two predicted requests (blue human = origin, red triangle = destination) and two vehicles (yellow car = origin, red triangle = destination of a passenger). Vehicle 1 has one passenger and vehicle 2 is empty. (b) Pairwise shareability RV-graph of requests and vehicles. Cliques of this graph are potential trips. (c) RTV-graph of candidate trips and vehicles which can execute them. A node (yellow triangle) is added for requests that can not be satisfied. (d) Optimized assignment given by the solution of the ILP, where vehicle 1 serves requests 2 and 3 and vehicle 2 serves requests 1 and 4. (e) Planned route for the two vehicles and their assigned requests. Vehicles that have no requests assigned at this stage either remain idle or are sent to rebalancing stations. This is computed using the rebalancer ILP. Taken from [2].

- The vehicle capacity is not exceeded
- A request drop-off never occurs before its pick-up
- The maximum delay and maximum waiting time of the requests is not to be exceeded

A schedule is vehicle specific, meaning that it is not necessarily true that a schedule that is feasible for one vehicle is also feasible for another. This is due to their different locations, passengers already on board, and due to potentially different vehicle capacities. In the implementation, a function $schedule(T, v)$ is defined which returns for a given trip T and a vehicle v according to the current state the optimal schedule and associated schedule cost. If no feasible schedule exists, it returns false.

In [2], this function is implemented using a backtracking algorithm. In this algorithm, candidate schedules for a trip are incrementally generated, adding schedule events to a schedule until all schedule events in a trip are added to the schedule. A candidate schedule is abandoned whenever one of the constraints cannot be met, or when the candidate schedule has a higher cost than a potentially already found full schedule. The cost function used is the sum of delays of all requests (including passengers) in the schedule.

To make the formulation more general, and to make it easier to allow for adding different constraints this function was also implemented as a mixed integer linear programming (MILP) based on vehicle routing problems (VRPs). This formulation and some results on the performance can be found in Appendix A. This was implemented in C++ using the state-of-the-art Gurobi solver. Although functional, this implementation surprisingly showed to be significantly slower compared to backtracking implemented in [2]. Therefore, a new backtracking algorithm was implemented similar to the work of [2] in Go.

3-3-2 Request-Vehicle Graph

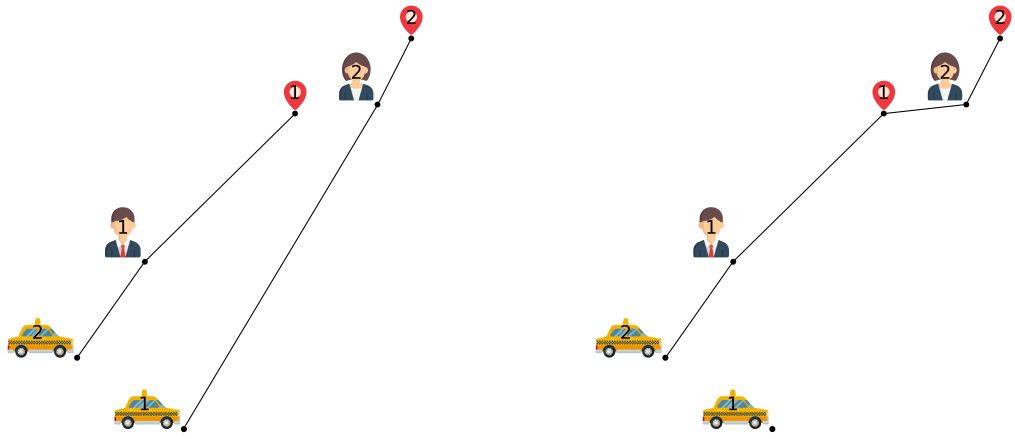
The RV graph is defined as an undirected graph $G = (V, E)$, in which a vertex is defined for every vehicle and request in $\mathcal{R}$ and $\mathcal{V}$. By the definition of a maximum waiting time, it is immediately possible to check which vehicles could potentially serve which requests and which cannot. To do so, the travel time of all vehicles to all requests are determined. An edge is added added between every vertex associated with vehicle v and request r for which $schedule(r, v)$ returns a feasible schedule. Notice that this is essentially a trip of size 1. Subsequently an edge is added between all vertices associated with request pairs r_i and r_j that can potentially be served together in a schedule by a single vehicle. This is tested by creating a virtual vehicle $v_{virtual}$ which is initialised at the location of r_i and r_j. If a feasible schedule can be computed for the virtual vehicle initialised either at r_i or r_j which serves both r_i and r_j, an edge is added between r_i and r_j.

3-3-3 Request-Trip-Vehicle Graph

Using the RV graph, an RTV graph is now constructed which contains trips larger than 1 request. The RTV graph is an undirected graph for which again a vertex is defined for all requests and vehicles. Subsequently a vertex is added for every trip for which a feasible

schedule exists for at least one vehicle. The vertex of this trip is connected to all vertices associated with requests within that trip, and all vertices associated with vehicles that can serve that trip. This is done by computing for every vehicle incrementally larger trips. Size 2 trips are constructed using all the individual requests that a vehicle can serve and by looking at which request can potentially be joined in a trip (from the RV graph). Subsequently, size 3 trips are formed from finding cliques in the trips of size 2 that are feasible for the particular vehicle. These steps are repeated up to trips of a maximum defined trip horizon $|T|_{max}$. In this process, for every feasible vehicle and trip combination the optimal schedule and the associated cost is stored.

Important to notice is that $|T|_{max}$ can be larger than the vehicle capacities. There are feasible schedules for trips that are larger in size than the remaining capacity of the vehicle, as long as not all pick-up events occur consecutively without any intermediate drop-offs. The potential benefit of allowing for trips larger than the remaining capacity is demonstrated in Figure 3-3. Increasing $|T|_{max}$ has a profound impact on the computation time however, since this increases the number of schedules that need to be considered. Making $|T|_{max}$ very small makes the schedules more myopic. The schedule generation however can only take into account requests that are placed currently. Since requests enter the system continuously, it might not necessary be very wise to compute very long schedules. The chance that a schedule will actually be executed until the end and not change as a result of a re-optimisation in a future time step is small. In this work, $|T|_{max}|$ is set at 4 regardless of total vehicle capacity and the remaining vehicle capacity. This is a rather arbitrary value, which was chosen as a tradeoff between fleet performance and computational time.



(a) Trip size limited to vehicle capacity. **(b)** No limit on trip size.

Figure 3-3: A comparison of schedules for two vehicles of capacity 1 with $|T|_{max} = 1$, and with $|T|_{max} = \infty$. The cost function in this case is the distance travelled by the vehicles. Two passengers, their associated destinations and two vehicles are placed in the plane. In the second case, the fact that the destination of passenger 1 is near to the origin of passenger 2 is used as an advantage. Both requests are handled by vehicle 2, because of which the total distance travelled by both vehicles is significantly smaller. This comes at a marginal increase of waiting time for passenger 2.

3-3-4 Trip-Vehicle Assignment

By the construction of the RTV graph, all feasible schedules are computed. It is now necessary to select the optimal assignment of vehicles to feasible schedules. This is performed using an ILP. In the original algorithm presented by [2], the cost that is minimised is the sum of delays of all requests. Any request that is not served and is ignored is penalised with a cost c_{ig}.

A set of binary decision variables $\epsilon_{i,j} \in \{0,1\}$ is defined for every edge between a vehicle j and a trip i in the RTV graph. An additional set of binary decision variables $\chi_k \in \{0,1\}$ is defined for $r_k \in \mathcal{R}$. The value of this variable is 1 if a request is not in any assigned schedule and is not to be served. Let ε_{TV} be the indices $\{i,j\}$ for all edges $e(T_i, v_j)$ in the RTV graph. Let $\mathcal{I}_j^V$ be the indices i for which an edge $e(T_i, v_j)$ exists, let $\mathcal{I}_i^T$ be the indices j for which an edge $e(T_i, v_j)$ exists, and let $\mathcal{I}_k^R$ be the indices i for which an edge $e(r_k, T_i)$ exists.

The total set of variables is then represented as: $\mathcal{X} = \{\epsilon_{i,j}, \chi_k \mid \forall i,j \in \varepsilon_{TV}, \forall r_k \in \mathcal{R}\}$. The cost function is given by:

$$\mathcal{C}(\mathcal{X}) = \sum_{i,j \in \varepsilon_{TV}} c_{i,j}\epsilon_{i,j} + \sum_{k \in \{0,\dots,n\}} c_{ig}\chi_k \tag{3-1}$$

In this cost function, $c_{i,j}$ is the cost of the assignment of v_j to T_i. Finally the ILP can be formulated as:

$$\min_{\mathcal{X}} \quad \mathcal{C}(\mathcal{X}) \tag{3-2}$$

subject to:

$$\sum_{i \in \mathcal{I}_j^V} \epsilon_{i,j} \leq 1 \qquad \forall v_j \in \mathcal{V} \tag{3-3}$$

$$\sum_{i \in \mathcal{I}_k^R} \sum_{j \in \mathcal{I}_i^T} \epsilon_{i,j} + \chi_k = 1 \qquad \forall r_k \in \mathcal{R} \tag{3-4}$$

$$\tag{3-5}$$

The number of binary variables in this ILP is equal to the number requests plus the number of edges between trips and vehicles in the RTV graph. In the worst case, all vehicles can serve all requests individually and all requests can be combined in trips. Although unlikely, this occurs when all vehicles and requests are close to each other and Ω and Δ are sufficiently large. The number of possible trip-vehicle pairs to explore in that case is equal to:

$$\sum_{i \in \{1,\dots,|T|_{max}\}} m\frac{n!}{i!(n-i)!} \tag{3-6}$$

In which n is the number of requests and m is the number of vehicles. In the worst case, the complexity of the trip-vehicle assignment is therefore $O(mn^{|T|_{max}})$.

3-3-5 Rebalancing

After the trip-vehicle assignment, often vehicles remain that are not assigned a schedule. This could be because there are no requests near the vehicle, or because other vehicles can serve all nearby requests at a lower cost. Instead of leaving these vehicles idle at their location, it might be beneficial for the fleet performance to send these vehicles to places where requests are to be expected in the near future. In the original algorithm presented in [2], idle vehicles are sent towards any requests that are ignored. Locations where requests are ignored have too few vehicles available, and so this is a simple way to redistribute vehicles towards regions where the supply of vehicles does not meet the demand. This however also means that vehicles are rebalanced after requests are already missed. It should therefore be possible to make a more effective rebalancing algorithm. Such an improved algorithm is presented in Chapter 4.

Chapter 4

Rebalancing

The text presented in this chapter is an adapted version of a paper that is under review at the time of writing of this thesis. The original version of the paper can be found in Appendix B. The author of this thesis worked together closely with authors of the named paper on the development of the codebase and implementation of the demand estimation and rebalancing algorithm.

4-1 Introduction

Besides the computation of efficient vehicle schedules, the proactive relocation of idle vehicles can have a significant influence on the fleet performance. Since the demand for vehicles is often not uniformly distributed, vehicles tend to build up in regions of low demand while vehicles are depleted in regions of high demand. For example in Manhattan, there are many trips to Harlem at night, but fewer back to Midtown in the morning. This mismatch in vehicle supply and demand means that vehicles often have to travel further than necessary to pick up customers, which leads to higher waiting times and more customer walk-aways. It also means that the number of passengers which a fleet can transport in a given time is less than optimal. Vehicle rebalancing focuses on positioning the idle vehicles so that future demand can be served with increased efficiency.

In this chapter the schedule optimisation algorithm presented in [2] and described in Chapter 3 is used and expanded with a method to continuously rebalance idle vehicles. A method is presented to determine an optimal discretisation of the operating area into well defined rebalancing regions. Furthermore a method is presented to estimate, from incoming transportation requests, a real-time demand per region. Using this estimation, subsequently the rebalancing of idle vehicles towards rebalancing regions is optimised. Finally, a case study is presented using real taxi data from Manhattan to demonstrate the benefits of the proposed rebalancer and to compare it to the previous state-of-the-art.

4-1-1 Contribution

Working further on the earlier work presented in [2] that computes an optimal assignment of a fleet of autonomous or human-driven vehicles to a set of requests, a method is presented that continuously rebalances the remaining idle vehicles over the operating area according to estimated real-time demands. This comprises:

- A method to optimally discretise the operating area into a set of rebalancing regions.
- A method to estimate vehicle demand for every rebalancing region using only the real-time request stream.
- An algorithm to assign idle vehicles to rebalancing regions using the estimated demand.
- Experimental validation comparing the performance of using no rebalancer, the rebalancer presented in prior work, and the new proposed rebalancing strategy.

4-2 Preliminaries

In this chapter, the same notations are used as introduced in Chapter 3. Furthermore, the operating area is partitioned into a set of rebalancing regions, $\mathcal{G}$, with region centres, $\mathcal{C}$ (described in Section 4-3-1). All locations closer to a region centre $c \in \mathcal{C}$ than any other region centre in $\mathcal{C}$ belong to the associated region.

4-2-1 Problem Formulation

After every predetermined time interval ψ, schedules are computed and assigned to vehicles so that the sum of delays of all requests is minimised. This step is referred to as the schedule assignment. For this the method presented in [2], and which is discussed in more detail in Chapter 3 is used. In some cases, not all vehicles in the fleet are assigned a schedule. This is either because there are no requests within a travel time smaller than Ω or because there are other vehicles available that are able to serve the requests more efficiently. This set of unassigned vehicles is denoted by $\mathcal{V}_r \subseteq \mathcal{V}$. These are the vehicles that are considered for rebalancing. In both cases, there is an oversupply of vehicles in those particular regions. At the same time, other regions might have an under supply of vehicles. In that case, requests in those regions might have to wait significantly longer before they are picked up and eventually might not be able to be serviced while respecting the constraints set by Ω and Δ.

The focus in this chapter is to determine how to distribute the unassigned vehicles over the operating area such that the request delay and waiting time is reduced, the number of ignored requests is minimised, and to do this dynamically over time.

4-2-2 Method Overview

The method to assign vehicle schedules and rebalance idle vehicles is split up into multiple steps. First, using an integer linear programming (ILP), the operating area is discretised into

Algorithm 1 Overview of the rebalancing method

$\mathcal{G} \leftarrow$ DiscretizeOperatingArea(t_{max})
for all $g \in \mathcal{G}$ **do**
  InitializeRateEstimate$_g$()
end for
for every time interval, ψ **do**
  $\mathcal{R} \leftarrow$ IncomingRequests()
  AssignVehicleSchedules($\mathcal{V}, \mathcal{R}$)
  $Q_g \leftarrow 0 \quad \forall g \in \mathcal{G}$
  for all $r \in \mathcal{R}$ **do**
    $g \leftarrow$ GetRebalancingRegion(r)
    $Q_g \leftarrow Q_g + 1$
  end for
  for all $g \in \mathcal{G}$ **do**
    UpdateRateEstimate$_g$(Q_g, ψ)
  end for
  $\mathcal{V}_r \leftarrow$ GetRebalancingVehicles()
  RebalanceVehiclesToRegions($\mathcal{V}_r, \mathcal{G}$)
end for

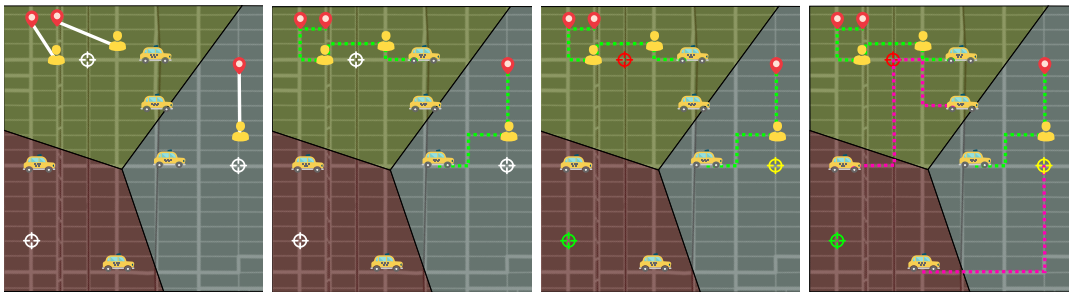
the set of regions $\mathcal{G}$ with region centres $\mathcal{C}$. The regions are computed once offline, and remain constant during the online schedule assignment. This process is explained in Section 4-3-1. A schematic overview of the steps performed at every assignment interval ψ is shown in Figure 4-1. The following steps are performed:

1. Vehicles are assigned schedules to pick up and drop off requests using the algorithm presented in [2].
2. Using the real-time request information, the current demand at every rebalancing region is estimated using a particle filter. See Section 4-3.
3. The vehicles that remained unassigned in the vehicle-schedule assignment (step 1) are assigned to rebalance towards regions in $\mathcal{G}$. This rebalancing assignment is computed using an ILP. See Section 4-4.

All vehicle schedules and rebalancing assignments are reconsidered at every assignment interval. Previously assigned vehicle schedules can change at each subsequent schedule assignment, and vehicles on the way to a rebalancing region can instead be assigned a schedule to pick up passengers. A detailed description of the method overview can be found in Algorithm 1.

4-3 Demand Estimation

During the execution of the algorithm, the rate is estimated at which incoming requests are being introduced at different locations in the operating area. This is done by first partitioning the operating area into a number of regions and for each region, utilising a particle filter to estimate the demand.



(a) Initial state **(b)** Schedule assignment **(c)** Demand estimation **(d)** Rebalancer assignment

Figure 4-1: Schematic overview of the method used for the assignment of vehicles to requests and the rebalancing of vehicles in the order as they are performed. (a) Example for a part of a road network with three regions (white marker = region centres), 5 vehicles, and three requests (yellow human = origin, red marker = destination). (b) Assignment of schedules to vehicles, the schedule trajectories are shown by the green dotted lines. Three of the five vehicles are not assigned a schedule. (c) The estimation of demands for every region according to the request information. In this figure, high demand is represented by the red marker, intermediate demand by the yellow marker and low demand by the green marker. (d) Optimised assignment of unassigned vehicles to rebalancing regions. The rebalancing vehicles move towards the region centres. The rebalancing trajectories are shown in the purple dotted lines.

4-3-1 Discretisation Into Regions

Given a directed graph, $G = (V, A)$, representing the road network where V is the set of vertices, and a matrix T where T_{ij} represents the travel time between vertex i and j, the problem is to select a subset of vertices $\mathcal{C} \subseteq V$ as region centres that can be used to aggregate demand. A request is in region $g \in \mathcal{G}$ if its origin is closer to the region centre c of region g than any other region centre in $\mathcal{C}$.

The operating area is discretised into regions using given parameter t_{max} which represents the maximum travel time between any vertex in the graph and the closest region centre. To determine the minimum number of regions for a given t_{max}, the problem is formulated as an ILP.

First a reachability matrix, R, is defined where $R_{ij} = 1$ if $T_{ij} \leq t_{max}$ and $R_{ij} = 0$ otherwise. This describes whether vertex j is reachable from vertex i given the time limit. Furthermore a set of binary variables x is defined where $x_i = 1$ if vertex V_i is used as a region centre and 0 otherwise. Using the reachability matrix and the binary variables, an ILP can be defined to determine the minimum number of region centres such that every vertex in the graph is reachable from at least one region centre as:

$$\min_{x} \quad \sum_{i=1}^{|V|} x_i \tag{4-1}$$

$$\text{s.t.} \quad \sum_{i=1}^{|V|} x_i \cdot R_{ij} \geq 1 \quad \forall j \in [1, |V|] \tag{4-2}$$

Constraint (4-2) ensures that every node in the road network graph is reachable within t_{max}

travel time by at least one region centre selected from the nodes in the graph. To extract the region centres, from V all vertices V_i are selected such that $x_i = 1$.

The region centres are computed a priori and are used to aggregate requests together so the rate of requests for each region can be computed. These region centres are also used for rebalancing as they are the locations that vehicles are proactively sent to.

4-3-2 Determining The Rate Of Requests

The vehicle demand is estimated online in each rebalancing region using only the real-time request stream. The vehicle demand is defined as the rate at which requests are originating from a given region over time.

The rate of requests at each region $g \in \mathcal{G}$ is modelled as an inhomogeneous Poisson point process with a stochastic time-varying rate, $\lambda_g(t)$. These rates are assumed to drift over a short time horizon according to a Wiener process. This means that for each region g, the change in rate of requests over time follows a Gaussian distribution, i.e. $\lambda_g(t') - \lambda_g(t) \sim \mathcal{N}(0, \nu \cdot (t' - t))$ for $t' > t$ and some given volatility parameter, ν. The rate, $\lambda_g(t)$, for each region is estimated using a sequential importance resampling particle filter as described in [35]. The particle filter is updated with the number of requests observed, n, within a time interval, $t - \epsilon_t$ to t. The N particles, $\{\hat{\lambda}_0^{(i)} : 1 \leq i \leq N\}$, are initialized uniformly at random within an given upper and lower bound at time 0. Their weights, $\{w_0^{(i)} : 1 \leq i \leq N\}$, are all set to $1/N$. The particles are updated in four steps.

1. N samples, $\{\hat{\lambda}_{t-\epsilon_t}^{(i)}\}$ with weights, $\{w_{t-\epsilon_t}^{(i)}\}$, are drawn with replacement from the particles with probabilities proportional to their weights.
2. Random noise is applied to each particle according to the Wiener process: $\hat{\lambda}_t^{(i)} = \hat{\lambda}_{t-\epsilon_t}^{(i)} + \epsilon_\lambda$, where $\epsilon_\lambda \sim \mathcal{N}(0, \nu \cdot \epsilon_t)$
3. The weights are updated with the observation of n requests in ϵ_t time: $\tilde{w}_t^{(i)} = w_{t-\epsilon_t}^{(i)} \cdot \Pr[k = n; \epsilon_t \cdot \hat{\lambda}_t^{(i)}]$, where $\Pr[k = n; \epsilon_t \cdot \hat{\lambda}_t^{(i)}]$ is the Poisson probability of n events with a rate $\epsilon_t \cdot \hat{\lambda}_t^{(i)}$
4. The weights are normalised: $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_k \tilde{w}_t^{(k)}}$

The estimate of the stochastic rate, $\lambda_g(t)$, for a region g at time t is then defined as the weighted average of the particles, $\lambda_g(t) = \sum_i w_t^{(i)} \cdot \hat{\lambda}_t^{(i)}$. The particle filter produces an estimate of the rate of requests for a given region by estimating the likelihood of a fixed number of candidate rates and returning the likelihood weighted average over the candidate rates.

4-4 Rebalancing

Due to the fact that demand is not equally distributed over the operating area, vehicles will tend to build up according to a spatial distribution that does not match the distribution of

demand. Due to this undesirable distribution of vehicles, it is possible that some vehicles remain idle while there are requests that are not served. This takes places when there are no vehicles that can reach these requests within the maximum waiting time Ω, or when the demand in a specific region is very high and there are not enough vehicles to serve all the requests in that specific region. In order to mitigate this problem, idle vehicles should be proactively rebalanced over the operation area so that their distribution matches the distribution of demand. This furthermore decreases waiting time since for incoming requests, the probability of having a nearby vehicle available is higher. A novel vehicle rebalancer is proposed that models the problem as an ILP to match the supply of vehicles to each area with the demand.

4-4-1 Implementation

The proposed rebalancer seeks to match the supply of idle vehicles in each region to the expected demand for a given time horizon $\mathcal{H}$. $\mathcal{V}_r \subseteq \mathcal{V}$ is defined as the set of idle vehicles that are available for rebalancing. These are the vehicles that are not assigned to pick up or drop off requests in the vehicle-schedule assignment. Furthermore $\mathcal{C}$ is defined as the set of region centres as described in Section 4-3-1. The goal is to find an assignment from vehicles in $\mathcal{V}_r$ to region centres in $\mathcal{C}$ such that the amount of requests the vehicles are able to serve is maximised, while not oversaturating or undersaturating regions with vehicles.

To solve this assignment, the problem can be formulated as an ILP. First a set of binary variables is defined as $\mathcal{X} = \{x_{ij} \mid \forall i \in [1, |\mathcal{V}_r|], \forall j \in [1, |\mathcal{C}|]\}$, where $x_{ij} = 1$ if vehicle i is assigned to rebalance to region centre j and zero otherwise. Also a travel time matrix T is defined, where T_{ij} is the travel time from vehicle i to the region centre j and a rate vector $\tilde{\lambda}$ where $\tilde{\lambda}_i$ is the current rate of requests at region i computed using the particle filter described in Section 4-3-2. With these variables, the objective function for the ILP which is to be maximised can be defined as:

$$\mathcal{J}(\mathcal{X}) = \sum_{i=1}^{|\mathcal{V}_r|} \sum_{j=1}^{|\mathcal{C}|} x_{ij} \cdot \tilde{\lambda}_j \cdot (\mathcal{H} - T_{ij}) \tag{4-3}$$

This objective represents the sum of the expected number of requests each vehicle would observe in its assigned rebalancing region for the given time horizon, $\mathcal{H}$. The expected number of requests observed by vehicle i is expressed as the rate of requests at the assigned region, $\tilde{\lambda}_j$, multiplied by the time remaining in the time horizon after the vehicle reaches the region, $\mathcal{H} - T_{ij}$.

A valid rebalancing assignment must guarantee that each vehicle is assigned to at most one station. This is described in the constraint:

$$\sum_{j=1}^{|\mathcal{C}|} x_{ij} \leq 1 \quad \forall i \in [1, |\mathcal{V}_r|] \tag{4-4}$$

Also, due to the formulation, the solution is constrained to assign vehicles to rebalancing regions that are reachable within the time horizon, $\mathcal{H}$, i.e. $\mathcal{H} \geq T_{ij}$. This constraint is formulated as:

$$x_{ij} \cdot (\mathcal{H} - T_{ij}) \geq 0 \quad \forall i \in [1, |\mathcal{V}_r|], \quad \forall j \in [1, |\mathcal{C}|] \tag{4-5}$$

In order to obtain an adequate dispersion of vehicles and limit the oversaturation of vehicles in rebalancing regions, the assignment needs to be constrained such that the supply of vehicles in a rebalancing region is less than some factor of their demand. The supply of vehicles in region $j \in [1, |\mathcal{C}|]$ for a given time horizon can be written as:

$$\sum_{i=1}^{|\mathcal{V}_r|} x_{ij} \cdot \frac{\mathcal{H} - T_{ij}}{\mathcal{H}} \tag{4-6}$$

The supply of vehicles is weighted by the percent of time in the next time horizon a vehicle would be able to sit idle at the assigned station. The time weighting is used to give a more accurate estimation of the vehicle supply. For example, if a vehicle takes 8 minutes to reach a region and the time horizon is set to 10 minutes, that vehicle's supply is only available for 20% of the time.

The demand for vehicles for some region $j \in [1, |\mathcal{C}|]$ and a given time horizon is defined as:

$$\tilde{\lambda}_j \cdot \mathcal{H} \tag{4-7}$$

Putting Equation (4-6) and (4-7) together a constraint can be formulated to limit the oversaturation of vehicles in rebalancing regions as:

$$\sum_{i=1}^{|\mathcal{V}_r|} x_{ij} \cdot (\mathcal{H} - T_{ij}) \leq \tilde{\lambda}_j \cdot \mathcal{H}^2 \cdot \rho \quad \forall j \in [1, |\mathcal{C}|] \tag{4-8}$$

Note that for a more concise description, the time horizon, $\mathcal{H}$, was multiplied on both sides of the inequality. Also note that a tuning parameter ρ was introduced that allows to specify an acceptable level of oversaturation at a rebalancing region.

Combining the objective function from Equation (4-3), $\mathcal{J}(\mathcal{X})$, with the constraints described in this section, an ILP can be formulated that finds an assignment of vehicles to rebalancing regions that maximises the expected number of requests observed by all vehicles while obtaining an adequate dispersion of vehicles to limit the oversaturation of vehicles in rebalancing regions. This ILP is then:

$$\begin{aligned} \max_{\mathcal{X}} \quad & \mathcal{J}(\mathcal{X}) \\ \text{s.t.} \quad & \text{constraints (4-4), (4-5), (4-8)} \end{aligned} \tag{4-9}$$

This optimisation will be executed repeatedly after every time interval ψ, after vehicles have been assigned schedules to pick up and drop off requests.

4-5 Evaluation

The proposed informed rebalancer is validated using historical taxi request data from Manhattan [34] and the performance is compared to the state-of-the-art. Since the rebalancing and scheduling algorithms use timeouts to prematurely exit from the optimisation, a more powerful computer can lead to much better results. To ensure a fair comparison to previous work, the rebalancer described in [2] was reimplemented and experiments were conducted on the same machine using the proposed informed rebalancer, the naive rebalancer from [2], without rebalancing and compared the performance of the Mobility-on-Demand (MoD) fleet.

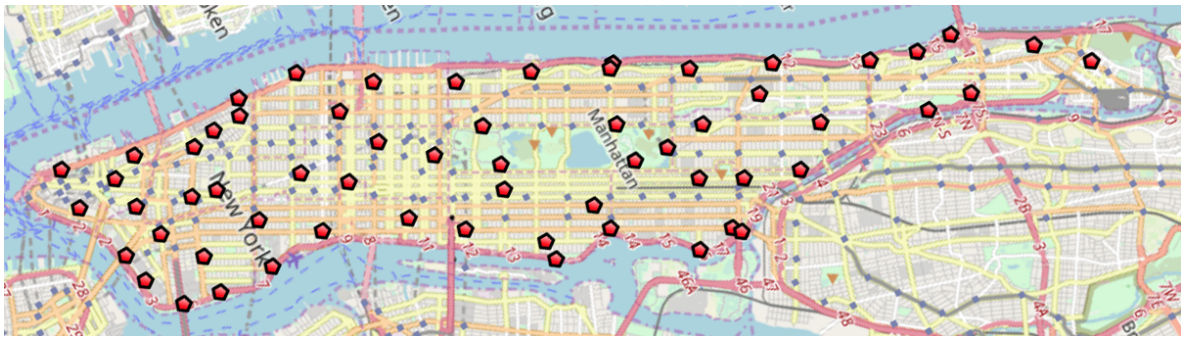
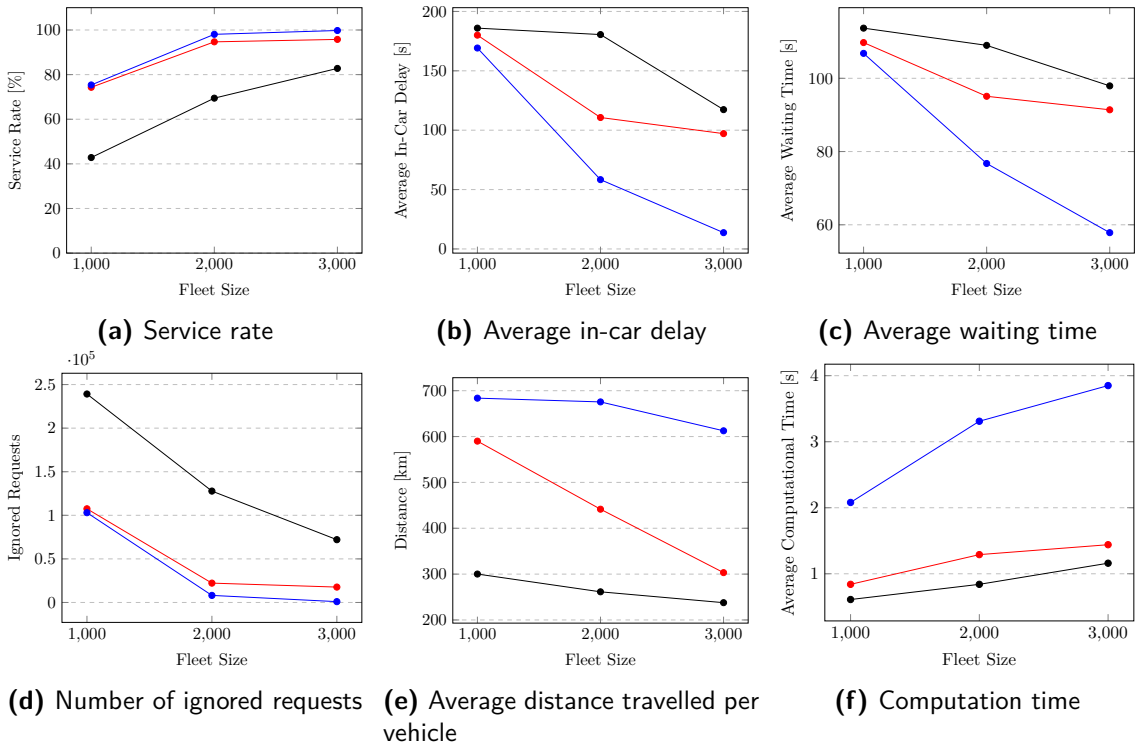


Figure 4-2: The computed location of region centres in Manhattan using the algorithm presented in Section 4-3 for a maximum reachability time $t_{max} = 150$ seconds.



(a) Service rate **(b)** Average in-car delay **(c)** Average waiting time

(d) Number of ignored requests **(e)** Average distance travelled per vehicle **(f)** Computation time

Figure 4-3: A comparison of several performance metrics for experiments with a fleet of 1000, 2000 and 3000 vehicles and no rebalancing, naive and informed rebalancing.

4-5-1 Experimental Setup

For the experiments one day of historical taxi data is used from 00:00 to 23:59 on May 1^{st}, 2013. This data is publicly available and contains all taxi trips in Manhattan [34]. The data contains the origin, destination, and associated pick up and drop off time for each taxi trip. As discussed in Chapter 3, it is assumed that the request time and pick up time are equal since the request time is not available. The experiments are executed using a simulator that

Fleet Size	Rebalancer	Avg. Delay [s]	Avg. Wait Time [s]	Avg. Dist. [km]	Avg. Comp. Time [s]	N. Ignored	% Serviced Reqs.
1000	No Rebalancing	185.94	113.70	300.14	0.61	239154	42.83
1000	Naive	180.03	109.76	590.02	0.84	107462	74.31
1000	Informed	169.20	106.81	683.75	2.08	103022	75.37
2000	No Rebalancing	180.53	109.00	261.39	0.84	127796	69.45
2000	Naive	110.68	95.09	441.57	1.29	22196	94.69
2000	Informed	58.36	76.74	675.55	3.31	8108	98.06
3000	No Rebalancing	117.32	97.93	237.72	1.16	72048	82.78
3000	Naive	97.11	91.40	303.24	1.44	17669	95.78
3000	Informed	13.72	57.85	612.56	3.85	964	99.77

Table 4-1: A detailed overview of the performance metrics for 1000, 2000 and 3000 vehicles for experiments with no rebalancer, and for the informed and naive rebalancer

simulates the movement of the vehicles, and to which requests are added according to the historical taxi data. The vehicle routes and travel times are determined using a stored road network of Manhattan. Like [2], travel time for each road segment are estimated using daily mean travel time computed by the method in [21] and a shortest path for every pair of nodes in the road network is pre-computed. A computer with a 2.6 GHz (overclocked to 4.0GHz), 18 core (36 threads) Intel i9 processor and 128GB of memory was used to run the experiments.

The performance is assessed for the rebalancing algorithm with a fleet size of 1000, 2000, and 3000 vehicles and a capacity of four passengers. A fixed maximum waiting time of $\Omega = 3$ minutes and maximum delay of $\Delta = 6$ minutes is used. All requests that cannot be served within these defined constraints on waiting time and delay are ignored (considered walk-aways), and dropped from the request pool. 100 particles are used to estimate the rate of requests in each region. The vehicle locations are initialised uniformly on vertices in the road network. The assignment interval was chosen as $\psi = 30$ seconds as in [2]. This means that vehicle schedules and the assignment of rebalancing stations are optimised every 30 seconds. At assignment time, all requests are considered that have not yet been picked up. To discretise the operating area into rebalancing regions, a maximum reachability time of $t_{max} = 150$ seconds is used which produced 61 regions. The centres of these regions are shown in Figure 4-2.

Two rebalancing techniques are evaluated: the proposed informed rebalancing algorithm and the naive rebalancing algorithm presented in [2]. The rebalancing algorithm in [2] assigns an idle vehicle to move to the locations of unassigned requests. The assignment minimises the sum of the distances travelled by the vehicles. The results of these rebalancing techniques are compared to a case were no rebalancing is performed.

4-5-2 Results

Several metrics are collected to assess the performance of the rebalancers including the service rate, in-car travel delay, waiting time, number of ignored requests, distance travelled per

vehicle, fleet utilisation, and computation time. The service rate is defined as the percentage of the total number of requests that were successfully served within the waiting time and delay time constraints. The in-car travel delay for a request is defined as $\delta_r - \omega_r$. The fleet utilisation is defined as the average percent of the fleet with assigned schedules throughout the day. The computational time includes the time required to compute schedules, estimate demands, and compute the rebalancing assignment. These metrics are plotted in Figure 4-3. The associated raw data is shown in Tab. 4-1.

It is observed that the service rate improves for all fleet sizes when using the proposed informed rebalancer rather than the naive rebalancer (See Figure 4-3a). Most notably, for a fleet size of 3000 vehicles, the service rate increases by 4%. Also the proposed rebalancer achieves a higher service rate with a fleet size of 2000 vehicles (98.1%) than the naive rebalancer with a fleet size of 3000 vehicles (95.8%). This means that by switching to the new proposed rebalancing algorithm, it is possible to reduce the fleet size by over 33% while maintaining the same service rate. Also a drastic reduction in walk-aways for all fleet sizes is observed (See Figure 4-3d). In particular, for a fleet size of 3000 vehicles, the proposed algorithm reduces the number of ignored requests by 95% compared to the naive approach.

The in-car travel delay and waiting time also benefit from informed rebalancing (See Figure 4-3b and 4-3c). For all fleet sizes, the average in-car travel delay and waiting time decreases when using the proposed rebalancer. For 3000 vehicles, the average delay drops from 97.1 to 13.7 seconds (86% improvement) and the average waiting time drops from 91.4 to 57.9 seconds (38% improvement).

Furthermore, a higher vehicle utilisation is observed for the informed and naive rebalancers compared to not rebalancing for all fleet sizes (See Figure 4-4). The informed rebalancer achieves the highest vehicle utilisation for all fleet sizes. The naive and informed rebalancers achieve similar utilisation for a 1000 vehicle fleet, but for 2000 and 3000 vehicle fleets, the informed rebalancer performs much better. This can be explained by the fact that both the naive and informed rebalancer for a 1000 vehicle fleet utilise almost all vehicles continuously over the duration of the experiment.

As in [2] and [33], it is observed that the advantages by using a rebalancer come at the cost of an increased distance travelled by the vehicles. This is apparent from Figure 4-3e. This might lead to higher fuel consumption, but the initial vehicle costs and costs of potential human drivers are much lower when using a smaller fleet with comparable performance. The reason for the larger travel distances is partly because more vehicles are being rebalanced and are moving when they are not assigned. This is also enforced however by the fact that the cost function used for assignment prefers using as many vehicles as possible with an as low as possible occupancy rate when feasible to serve requests to minimise the delay. This cost does not take into account the collective distance travelled by the vehicles.

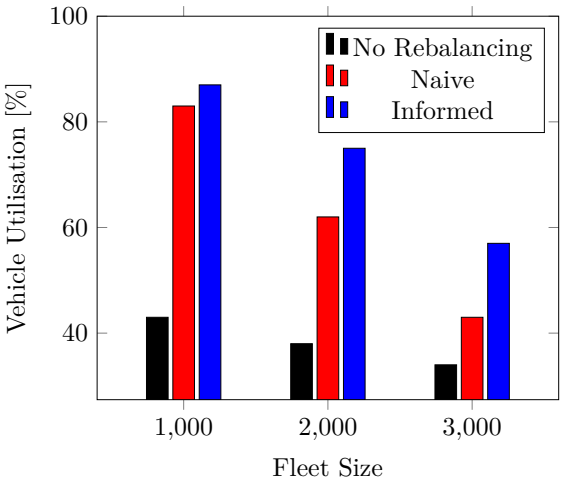


Figure 4-4: Vehicle utilisation for 1000, 2000, and 3000 vehicle fleet sizes. The vehicle utilisation is defined as the average percent of vehicles assigned to trips over the entire experiment. For each fleet size, the vehicle utilisation is measured without rebalancing, using the naive rebalancer, and using the proposed informed rebalancer.

Chapter 5

Fleet Composition

In the experiments presented in the previous chapter, the fleet comprised vehicles of capacity 4 exclusively. This is motivated by the fact that this corresponds to the capacity of regular taxi vehicles. In current public transportation systems however, often a significant part of urban transportation on roads is served by high capacity buses. These buses generally operate on static routes which are preplanned and do not take into account actual demand. The goal is to see if this can be replaced by buses that serve transportation requests according to dynamic routes which are computed in a similar way as already demonstrated for capacity 4 vehicles. In such a system, high-capacity buses could serve routes where demand is high, while regular vehicles serve requests in areas of lower demand where a bus would be inefficient. For this purpose, the effect of introducing high capacity vehicles to a fleet is investigated using the currently implemented routing and rebalancing algorithm. The implications on fleet performance are studied for both homogeneous fleets of high capacity vehicles as well as heterogeneous fleets comprising both buses and regular vehicles.

5-1 Method

In order to study the effect of introducing buses to the fleet, the same implementation is used as presented in Chapter 4 on the Manhattan taxi data set. A fixed fleet size of 1000 vehicles is chosen, since the number of served requests for a fleet of 2000 and 3000 vehicles is already close to 100% even for a fleet of regular sized vehicles (from the results presented in Section 4-5-2). A fleet of 1000 vehicles of capacity 4 using this implementation is only capably of serving roughly 75% of the requests. This means that there is still room for improvement, and that possible improvements will be visible in the results.

Other than their capacity, vehicles of different capacities are assumed to have exactly the same characteristics. In order to test the effect of larger vehicle capacities on fleet performance, the performance is first tested for several capacities of vehicles in homogeneous fleet of size 1000. This allows to identify if and to what extend larger vehicle capacities have an influence on fleet performance. From these results, an optimal bus capacity can be selected.

Subsequently, for a fleet of 1000 vehicles, experiments are conducted with varying ratios of vehicles of two different capacities. Part of the fleet will consist of regular sized capacity 4 vehicles, and part of the fleet will consist of larger capacity buses. These experiments allow to identify for the current implementation how vehicles of different capacities in a single fleet cooperate, and what the effect is of different ratios of both vehicle types on fleet performance.

5-2 Results

In order to asses the fleet performance, several service metrics are collected. Specifically the service rate, average in-car delay, average waiting time, number of ignored requests and the average distance travelled per vehicle are recorded for a simulation of 24 hours starting and ending at midnight. These metrics were defined in Chapter 4. The experiments are conducted under similar circumstances and with the same dataset as used in Chapter 4.

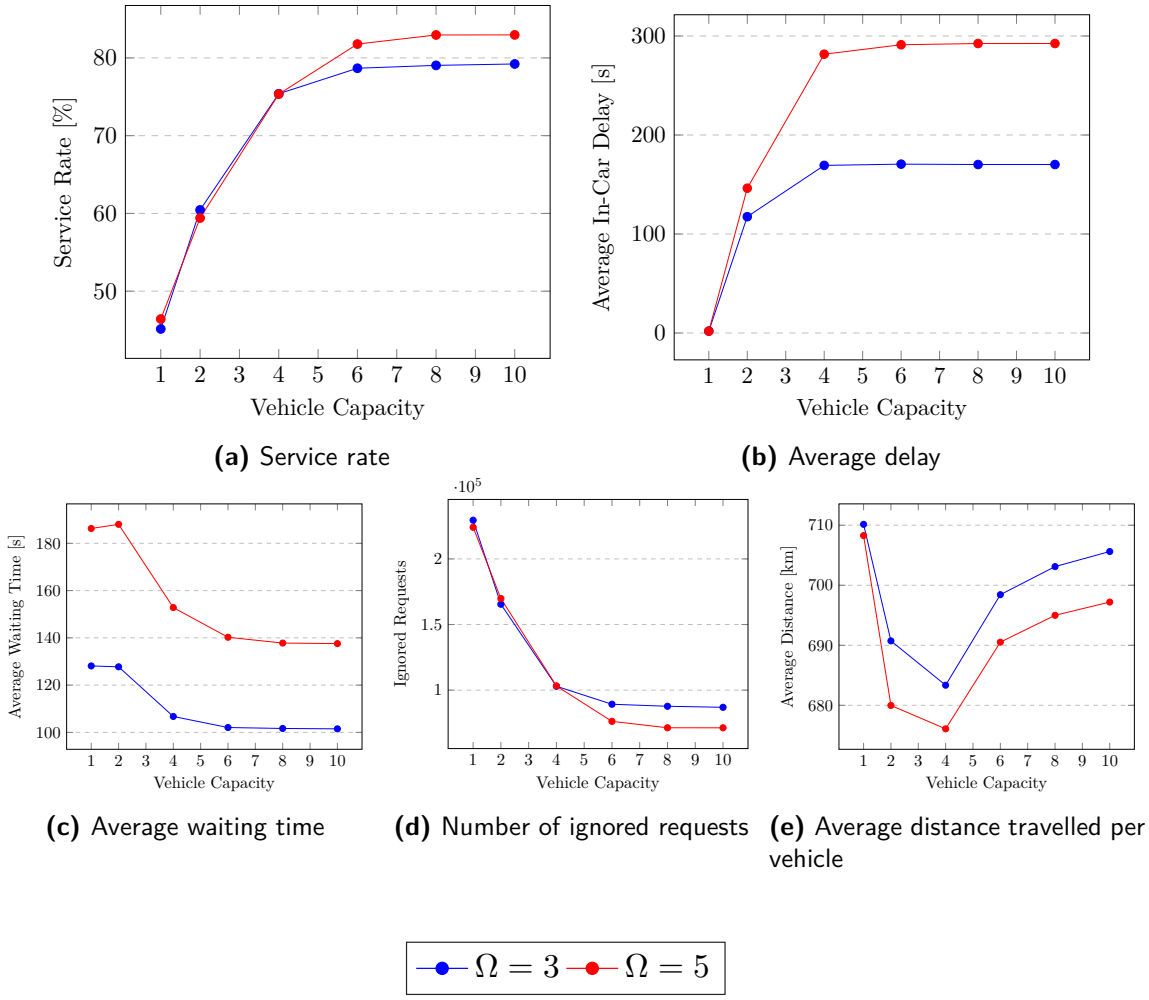
5-2-1 Homogeneous Fleet

Simulations are conducted for a homogeneous fleet of 1000 vehicles with capacities varying from 1 to 10 seats. Results are collected for a maximum waiting time of $\Omega = 3$ and $\Omega = 5$ minutes, with a maximum delay of $\Delta = 2\Omega$. The results of these experiments are shown in Figure 5-1, and a detailed overview of the metrics is shown in Tables 5-1 and 5-2.

From the results, it can be seen that for both time constraints the effect of increasing vehicle capacity is largest at low vehicle capacities. Increasing the vehicle capacity from 1 to 4 seats leads to a significant increase in service rate (and a decrease in number of ignored requests) from approximately 45% to 75% for both time constraints. Furthermore, a significant decrease in waiting time and increase in average in-car delay is observed. The average distance travelled fluctuates slightly around 700 km, and is minimised for capacity 4.

For capacities over 6, all performance metrics stay relatively constant. The service rate settles around 79% for a maximum waiting time of 3 minutes, and at 83% for a maximum waiting time of 5 minutes. The waiting time and in-car delay for a maximum waiting time of 3 minutes settles at approximately 170 seconds and 100 seconds respectively. For the experiments with a maximum waiting time of 5 minutes, the waiting time settles at 137 seconds and the in-car delay settles at 292 seconds. For both time constraints, the average delay (which is the sum of the in-car delay and waiting time) approaches the maximum defined delay of 6 and 10 minutes. In the experiments with $\Delta = 6$ minutes, the average delay settles at approximately 4.5 minutes while in the experiments with $\Delta = 10$ minutes, the average delay settles around 7.2 minutes.

Furthermore, the difference in service rate between the experiments for $\Omega = 3$ and $\Omega = 5$ for experiments up to capacity 4 is small. The average waiting time and delay for the experiment with a maximum waiting time of 5 minutes however is significantly higher compared to the experiments with a maximum waiting time of 3 minutes. An additional experiment was conducted with a fleet of 1000 vehicles of capacity 4 with $\Omega = 7$ minutes and $\Delta = 14$ minutes. Compared to the experiments with the same fleet but with a lower maximum waiting time and delay, this degrades fleet performance even further. These time constraints yield a service rate of 71.28% (4.1% lower than with $\Omega = 3$ minutes), with a significantly higher average waiting time and in-car delay of 187.60 seconds and 375.04 seconds respectively.



(a) Service rate

(b) Average delay

(c) Average waiting time

(d) Number of ignored requests

(e) Average distance travelled per vehicle

Figure 5-1: A comparison of several performance metrics for experiments with a homogeneous fleet of 1000 vehicles with different vehicle capacities. Results are shown for $\Omega = 3$ minutes and $\Omega = 5$ minutes. In the experiments $\Delta = 2\Omega$.

5-2-2 Heterogeneous Fleet

Results are collected for a heterogeneous fleet of vehicles of regular sized vehicles of capacity 4 and buses of capacity 8. The fleet size is 1000 vehicles in all experiments, but the ratio of buses to regular vehicles is varied from 0 (no buses) to 1 (only buses). The performance metrics of the experiments are shown in Figure 5-4, and the corresponding detailed overview is shown in Table 5-3. Furthermore a plot of the number of vehicles with more than 4 passengers on board in a fleet of 1000 buses of capacity 8 over time is shown in Figure 5-2. A plot of the number of vehicles utilised per type for passenger transport over time in a fleet of 500 buses of capacity 8 and 500 regular vehicles is shown in Figure 5-3.

From Figure 5-4 it can be seen that the service rate increases approximately linearly with an increased fraction of vehicles of capacity 8. The service rate varies from 75.31% with a fleet of only regular vehicles, to 82.82% with a fleet of only buses. The waiting time and in-car delay both do not change dramatically. The waiting time and in-car delay change respectively from

Vehicle Capacity	Average In-Car Delay [s]	Average Waiting Time [s]	Average Distance [km]	Number Ignored	Serviced Requests [%]
1	1.93	128.12	710.15	229487	45.14
2	117.49	127.75	690.73	165453	60.45
4	169.3	106.74	683.35	102954	75.39
6	170.57	102.06	698.43	89251	78.67
8	170.19	101.66	703.107	87702	79.04
10	170.19	101.50	705.62	86915	79.22

Table 5-1: A detailed overview of the performance metrics for a homogeneous fleet of 1000 vehicles with different vehicle capacities with $\Omega = 3$ minutes and $\Delta = 6$ minutes.

Vehicle Capacity	Average In-Car Delay [s]	Average Waiting Time [s]	Average Distance [km]	Number Ignored	Serviced Requests [%]
1	1.93	186.27	708.05	224127	46.42
2	146.26	188.03	679.98	169799	59.41
4	281.61	152.82	676.08	103157	75.34
6	291.14	140.25	690.52	76208	81.78
8	292.39	137.80	694.99	71318	82.95
10	292.40	137.56	697.20	71287	82.96

Table 5-2: A detailed overview of the performance metrics for a homogeneous fleet of 1000 vehicles with different vehicle capacities with $\Omega = 5$ minutes and $\Delta = 10$ minutes.

approximately 153 seconds and 282 seconds to 138 seconds and 292 seconds. It is seen that the performance metrics of the heterogeneous fleet are bounded by the performance metrics obtained with a homogeneous fleet of vehicles of capacity 4 and 8.

From Figure 5-2 it can be observed that many vehicles have more than 4 passengers on board from 08:00 to approximately 16:00 and from approximately 17:00 until midnight. The times of high numbers of vehicles with more than 4 passengers on board starts roughly at the morning and evening rush. From midnight until before the morning rush, barely any vehicles have more than 4 passengers on board. The maximum number of vehicles with more than 4 passengers on board during the experiment is 423, this occurs between 19:00 and 20:00.

Figure 5-3 shows the number regular vehicles and buses of capacity 8 in use for passenger transportation. It can be seen that almost all vehicles in the fleet are either moving towards a pick-up, or have passengers on board during most of the day. Only during the middle of the night, from 00:00 to 7:00, there are a large number of vacant vehicles. It can be observed that throughout the simulation, the number of vehicles of both types that are used is always approximately equal.

Additionally, an experiment was conducted with the same implementation, but where vehicles of different capacities have a different maximum waiting time and maximum delay for the passengers travelling on these vehicles. In this experiment, for buses of capacity capacity 8

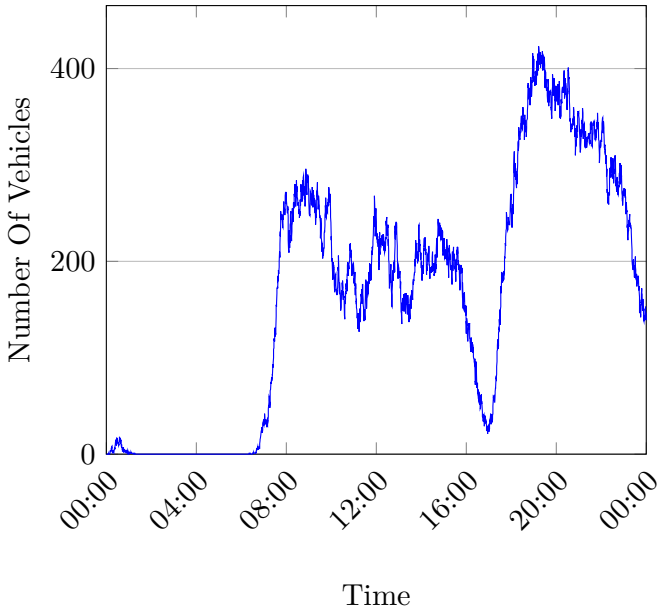


Figure 5-2: Number of vehicles with more than 4 passengers on board for a fleet of 1000 vehicles of capacity 8 for an experiment of 24 hours starting at midnight with $\Omega = 5$ minutes and $\Delta = 10$ minutes.

$\Omega = 5$ minutes and $\Delta = 10$ minutes. For regular vehicles of capacity 4 in this case $\Omega = 3$ minutes and $\Delta = 6$ minutes. For a fleet of 1000 vehicles with 500 buses and 500 regular vehicles, this yields a service rate of 79.87%, an average in-car delay of 239.87 seconds and an average waiting time of 124.43 seconds. This is a slightly higher service rate with a significantly lower average waiting time and delay compared to an experiment with the same fleet, but where $\Omega = 5$ and $\Delta = 10$ for all vehicles.

5-3 Discussion

From the experiments with different vehicle capacities in a homogeneous fleet, the most important observation is that increasing vehicle capacity only has a beneficial effect on the number of served requests up to a certain capacity. It can be seen that average waiting time decreases with increased vehicle capacity. The decrease in waiting time can be explained by the fact that with high capacity vehicles, it is more likely that a vehicle close to a new request still has a vacant seat available and can pick the request up quickly. The average passenger delay however increases with increasing vehicle capacity. Combining multiple requests in a single vehicle generally introduces a larger delay compared to serving them individually due to the detours required to serve all of the passengers in a combined schedule. This explains the fact that more passengers can be served with a fleet of high capacity vehicles, but at the cost of a higher average delay. This however also explains why the increase in service rate with increasing vehicle capacity is bounded. Apparently it is rare to find vehicle schedules in the Manhattan dataset in which more than 8 passengers are in the same vehicle at the same time while all passengers have a delay lower than 10 minutes. This motivates the decision to limit the capacity of buses in a fleet to capacity 8.

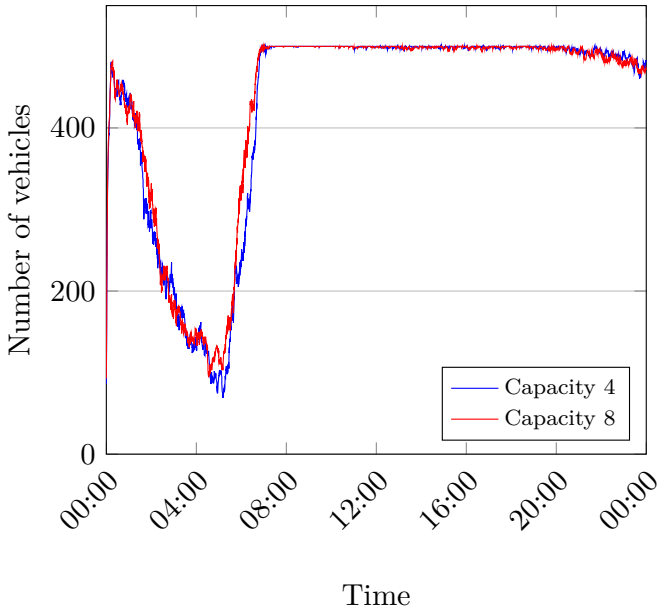
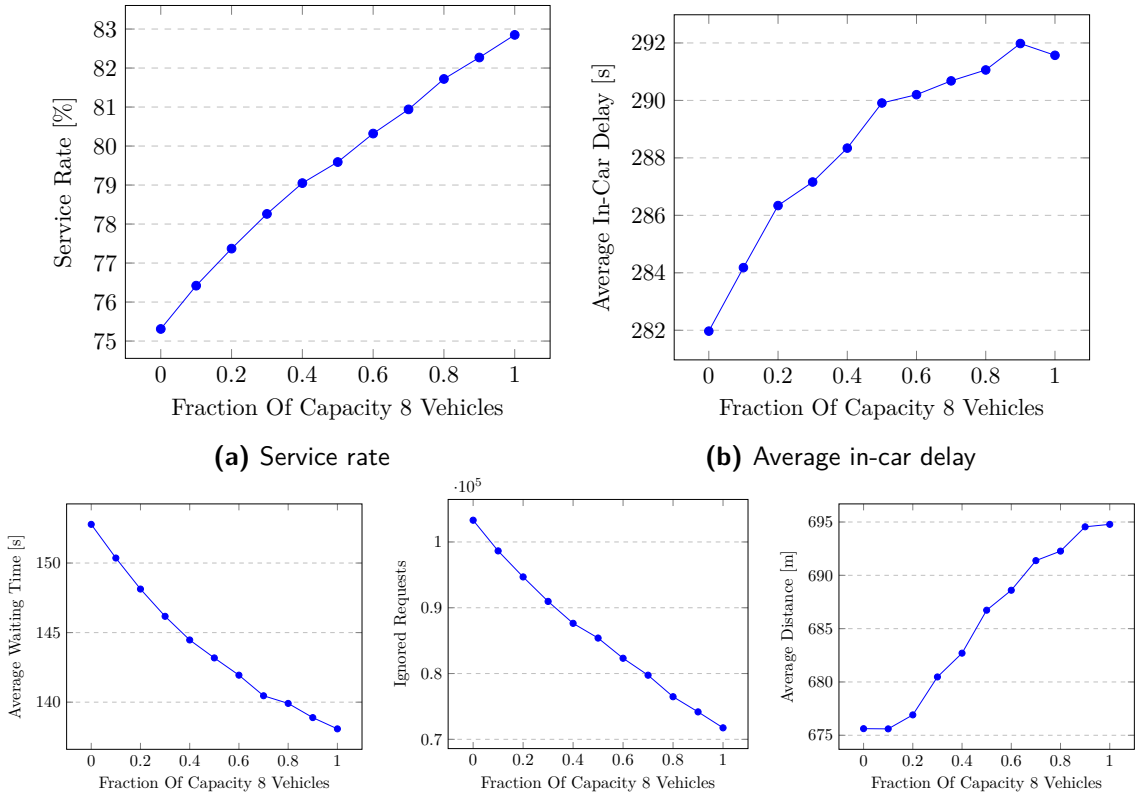


Figure 5-3: Number of vehicles of capacity 4 and capacity 8 that are assigned to schedules in a fleet of 500 vehicles of capacity 4 and 500 vehicles of capacity 8 for an experiment of 24 hours starting at midnight with $\Omega = 5$ minutes and $\Delta = 10$ minutes. Vehicles are assigned schedules when they are either moving towards a request pick-up or are transporting passengers.

Furthermore it was found that increasing the maximum waiting time and delay does not necessarily lead to an increase in service rate, while it does generally lead to an increased average waiting time and delay. For a fleet of 1000 vehicles of capacity 4, it was found that the fleet performs best for $\Omega = 3$ and $\Delta = 6$. When this is increased to $\Omega = 7$ and $\Delta = 14$, this fleet performs significantly worse in terms of service rate, average waiting time and average delay. This is most likely due to the fact that a large part of the vehicle seats are already utilised during the experiment with this limited fleet size even with a low maximum waiting time and delay. Increasing the maximum waiting time and delay initially allows for more requests to be served. Since the cost of ignoring a request is high, schedules with higher request delays that are now feasible are also assigned. Therefore requests on average occupy vehicle seats for a longer time. This inhibits the vehicle's ability to serve future requests due to its limited capacity. Since the vehicle capacity was already effectively fully utilised even for lower time constraints for this fleet, this leads to a degradation in fleet performance.

The experiments with a heterogeneous fleet of capacity 8 buses and regular sized vehicles demonstrate how vehicles of different types cooperate in a fleet with the current method for vehicle routing. The performance metrics change approximately linearly when introducing more buses to a fleet of a fixed number of vehicles from 0% buses to 100% buses. In all experiments, the number of buses and regular vehicles utilised for passenger transport is always approximately equal in ratio to the number of vehicles of both types in the fleet, regardless of demand. It seems therefore that there is no bias towards the utilisation of a certain type of vehicle at any time. It is furthermore apparent that the buses are used ineffectively. The number of buses that have more than 4 passengers on board (and could therefore not be served by a regular vehicle) is always relatively low. The highest number is obtained in simulations for a fleet of 1000 buses, in which at the busiest time less than half



(a) Service rate **(b)** Average in-car delay

(c) Average waiting time **(d)** Number of ignored requests **(e)** Average distance travelled per vehicle

Figure 5-4: Performance metrics for a 24 hour simulation for a fleet of 1000 vehicles of capacity 4 and capacity 8 in varying ratios with the delay schedule cost function. In the experiments $\Omega = 5$ minutes and $\Delta = 10$ minutes.

of the buses have more than 4 passengers on board. This indicates that in theory, it could be possible to get the same service rate of a homogeneous fleet of 1000 buses with a fleet consisting of closer to 500 buses and 500 regular vehicles. This would mean that most buses are used for schedules in which the capacity of regular vehicles would have been exceeded. In that case, the performance metrics should change steeply when increasing the number of buses from 0 to 500, while it remains approximately constant when increasing the number of buses from 500 to 1000. In practice this is difficult to obtain, since it is not known where these schedules occur in the operating area, and if these can be served by the same vehicles.

Finally, it was shown that for performance of a heterogeneous fleet it can be advantageous to define a different maximum waiting time and delay for buses and regular sized vehicles. As demonstrated in the experiments with a homogeneous fleet, it can be detrimental for service rate, average waiting time and average delay to increase the maximum waiting time and delay when fleet size and vehicle capacity is limited. Whether increasing the maximum waiting time and delay is beneficial depends however on vehicle capacity. For a fleet of 1000 buses of capacity 8, among the conducted experiments the best service rate is obtained for $\Omega = 5$ and $\Delta = 10$. For a fleet of 1000 regular vehicles of capacity 4, the best fleet performance is

Vehicles of Capacity 8	Average In-Car Delay [s]	Average Waiting Time [s]	Average Distance [km]	Number Ignored	Serviced Requests [%]
0	281.97	152.78	675.62	103305	75.31
100	284.18	150.36	675.60	98641	76.42
200	286.34	148.13	676.91	94686	77.37
300	287.16	146.17	680.47	90963	78.26
400	288.34	144.47	682.70	87631	79.05
500	289.91	143.18	686.73	85393	79.59
600	290.2	141.94	688.60	82324	80.32
700	290.68	140.46	691.38	79749	80.94
800	291.06	139.91	692.27	76476	81.72
900	291.98	138.89	694.55	74170	82.27
1000	291.57	138.08	694.78	71747	82.85

Table 5-3: A detailed overview of the performance metrics for a heterogeneous fleet of 1000 vehicles with capacity 4 and 8 in varying ratios with $\Omega = 5$ minutes and $\Delta = 10$ minutes.

obtained for $\Omega = 3$ and $\Delta = 6$. Consequently, for a fleet of 500 buses and 500 regular vehicles a higher service rate, lower average waiting time and lower average delay is obtained when $\Omega = 3$ and $\Delta = 6$ for regular vehicles, and $\Omega = 5$ and $\Delta = 10$ for buses compared to when $\Omega = 5$ and $\Delta = 10$ for all vehicles.

Chapter 6

Demand Predictive Schedule Assignment

From the results presented in the previous chapter, it seems that buses in a heterogeneous fleet can be used more effectively. In the implementation used so far, vehicle routes are optimised based on requests currently placed. The vehicle-schedule assignment does not anticipate requests that will be placed in the future. By using estimates on future demand per region in the operating area, the assignment of vehicle routes could be changed so that vehicles are positioned more optimally to serve future requests. In this chapter a model is proposed to optimise the assignment of schedules to vehicles while taking into account the vehicle positioning for serving future requests. This should also help to route buses towards areas of high demand, where their extra capacity can be most effectively utilised.

6-1 Motivation

There are several reasons to take into account predicted future requests in the assignment of schedules to vehicles. In the previous chapter it could be seen that buses were not utilised effectively. A limited fraction of buses was serving schedules that could not be served by regular vehicles even in case of an undersupply of vehicles. Buses are most efficient when they are driving in areas of high demand, where the probability is high that multiple requests can be combined in one vehicle. By taking into account predicted future requests, these buses can be assigned schedules that take the buses into areas of high demand.

The advantage is not only limited to buses however. Even for regular vehicles it can be beneficial to favour schedules that take a vehicle through areas of high demand instead of low demand. When a vehicle can choose between two similar schedules with approximately the same schedule costs, it is almost always favourable to choose the schedule that leads the vehicle through the areas of highest demand. This will maximise the chance of being as close as possible to future request and thus being able to serve these future request as efficiently as

possible. Sometimes it might even be desirable to select schedules with substantially higher costs if these schedules improve the position of vehicles for serving future requests.

A simple example is illustrated in Figure 6-2. In this example it might be better to assign the blue schedule than the orange schedule although the latter might have a higher cost. Since the blue schedule leaves the vehicle in an area of high predicted demand, and the orange schedule in a region of low demand it is more likely that the vehicle will be closer to future placed requests and will be able to serve those faster compared to when it follows the orange schedule. Furthermore, during both schedules the vehicle has a remaining capacity of 3 seats for most of the schedule. Therefore, requests that might appear while this schedule is served might potentially also be picked up immediately. Such pick-ups are more likely with the blue schedule than the orange schedule since the blue schedule traverses an area of higher demand.

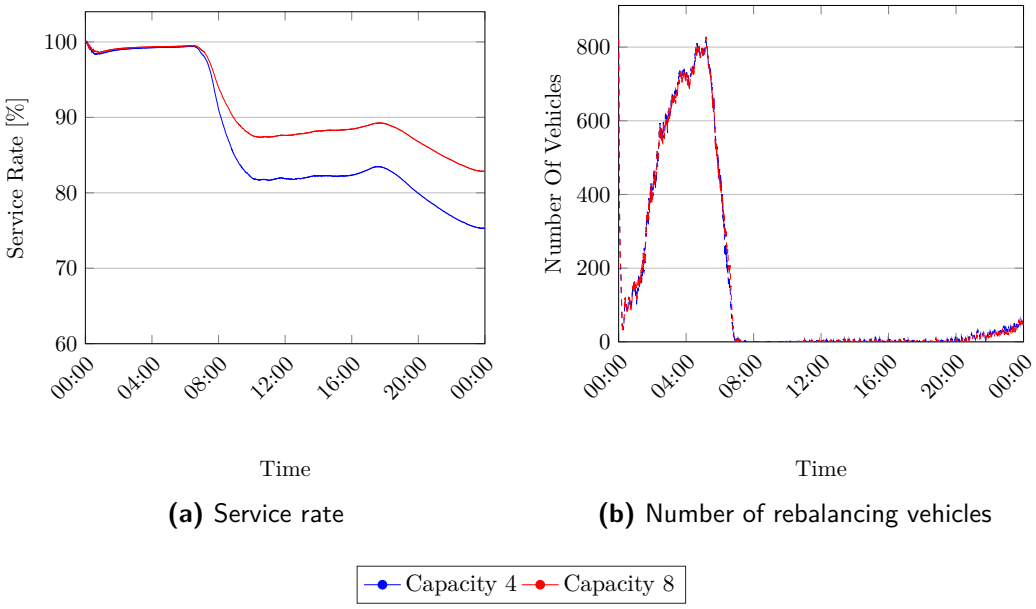
It seems that this issue could also be addressed by simply rebalancing idle vehicles to the areas of high demand, to make sure there are always vehicles available in the busy areas and to make sure that buses are specifically sent to areas of high demand. In the previous experiments however it is observed that for a limited fleet size of 1000 vehicles, service rate decreases at the most busy times. At these moments virtually all vehicles are assigned to schedules and none of them are rebalancing. This can be verified in Figure 6-1, which shows the service rate and the number of rebalancing vehicles for two different experiments for a full day. It is therefore expected that changing the rebalancing strategy will have little effect on fleet performance for a limited fleet size. This is supported by the results found in Chapter 4 as well. For a fleet of capacity 4 vehicles, large improvements in service rates were obtained for fleets with 2000 and 3000 vehicles with the new rebalancer, but a limited increase in service rate was obtained for a fleet of 1000 vehicles. The focus is therefore on improving the vehicle-schedule assignment.

6-2 Method

In this section a method is described that attempts to optimise the vehicle-schedule assignment so that requests currently in the system are served efficiently, while vehicles are also positioned favourably for serving future requests. This method computes two schedule costs: a cost which expresses how efficiently requests currently in the system are served, and a cost which expresses how well the schedule positions the serving vehicle for future requests. The assignment of schedules to vehicles is performed using an integer linear programming (ILP) with a cost function expressing the trade-off between schedule value based on current requests, and anticipated future requests. The method can be split into several steps that are performed at every assignment time ψ. These steps are described in detail in Algorithm 2. Briefly, these steps are:

1. The current demand is estimated for every region $g \in \mathcal{G}$ in the operating area according to the method presented in Chapter 4.

2. Feasible schedules are computed for all vehicles $v \in \mathcal{V}$, and the cost of each schedule according to the requests currently in the systems is stored. This is done using the same method as presented in Chapter 3.



(a) Service rate **(b)** Number of rebalancing vehicles

Figure 6-1: Service rate and vehicle status over time for an experiment with a homogeneous fleet of 1000 vehicles of capacity 4 and capacity 8. The experiment runs for 24 hours starting at midnight. The number of rebalancing vehicles for both experiments is similar and largely overlaps in the plot.

3. A cost is computed for every feasible schedule expressing how well it enables the vehicle to serve future predicted demand. This cost is computed according to the model described in Section 6-2-1.

4. Schedules are assigned using an ILP with a cost function expressing the trade-off between the cost of the schedule for serving current requests and the cost of the schedule in anticipation to future requests.

5. Remaining unassigned vehicles are assigned to rebalance towards regions $g \in \mathcal{G}$ according to estimated demand using an ILP. The rebalancing is performed using the same method as described in Chapter 4.

6-2-1 Anticipated Future Demand Schedule Cost

In this section a cost model is described which is used to evaluate the quality of a schedule for serving predicted future requests. This model predicts the number of additional future requests a vehicle serving a certain schedule can pick up while executing the schedule. It is difficult to make a model that accurately predicts this since it depends on a lot of factors. The goal however is to make a simple model which demonstrates that such an approach can yield a benefit for fleet performance. The number of expected future pick-ups in a schedule is assumed to depend on the number of vacant seats in a vehicle, the passengers already on board of the vehicle, and the demand in the regions that the vehicle traverses. For a given schedule, the number of expected pick-ups is computed for every subsection of the schedule and summed up to a predefined horizon of 15 minutes. A subsection of the schedule is the

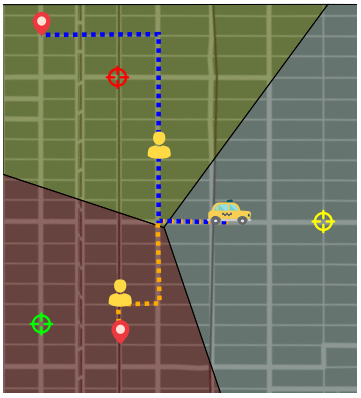


Figure 6-2: A situation where one empty vehicle of capacity 4 is available and two requests appear at the same time. The request origins are represented by the yellow humans, and their destinations are represented by the red markers. The road map is split into three regions with region centres represented by the crosshairs. Their colours represent the demand in those regions (green = low demand, yellow = medium demand and red = high demand). The dashed orange and blue lines represent two possible schedules for the vehicle. The vehicle has to choose between serving one of the two requests.

route between every pair of subsequent schedule events. This includes the subsection from the vehicle's current location to the first schedule event. If a total schedule duration is less than 15 minutes, the expected number of pick-ups are computed and added as if the vehicle would remain idle in the region of the last event. A detailed overview of the method is given in Algorithm 3. The number of expected pick-ups between two schedule events is defined as:

$$\pi = n_e \cdot \rho_l \cdot \rho_s \tag{6-1}$$

In which n_e is the number of empty seats on the vehicle (which is constant between two schedule events), ρ_l is a correction factor for the number of passengers already on board, and ρ_s is a correction factor for the demand in the region which the schedule traverses. The computation of the correction factors will be discussed in the subsequent sections.

Demand

The number of expected additional pick-ups of a vehicle moving according to an assigned schedule most importantly depends on the demand in the regions that it traverses during schedule execution. The demand in regions in the operating area is computed in real-time as described in Chapter 4. The demand is computed for all events in the schedule, according to the region that this event is in. To approximate the demand in the route between two events, the average is taken of the demand at the two adjacent events. The number of seen requests during the execution of a section of the schedule is defined as the average demand multiplied by the duration of that schedule section.

The larger the number of seen requests, the larger the probability that among these seen requests, there are requests that the vehicle can pick up. This is represented by a seen request multiplier. This multiplier is equal to one for infinitely many seen requests. The

Algorithm 2 Overview schedule assignment with current schedule cost and anticipated future request schedule cost.

for every time interval ψ **do**
  $\mathcal{R} \leftarrow$ IncomingRequests()
  $\mathcal{G} \leftarrow$ UpdateRateEstimate($\mathcal{R}$)
  $\mathcal{S} = \emptyset$
  for all $v \in \mathcal{V}$ **do**
    $\mathcal{S}_v \leftarrow$ ComputeFeasibleSchedules($v, \mathcal{R}$)
    $\mathcal{S} \leftarrow \mathcal{S}_v$
    for all $S \in \mathcal{S}_v$ **do**
      $\pi_S \leftarrow$ EstimateAdditionalPickups(v, S)
    end for
  end for
  AssignVehicleSchedules($\mathcal{V}, \mathcal{S}, \pi$)
  $\mathcal{V}_r \leftarrow$ GetRebalancingVehicles()
  RebalanceVehiclesToRegions($\mathcal{V}_r, \mathcal{G}$)
end for

probability is then 100% that the vehicle will be able to fill all its seats with passengers in that section. It is defined as:

$$\rho_s = \frac{\mathcal{R}_{seen}}{\mathcal{R}_{seen} + \alpha} \tag{6-2}$$

In this equation, α is a model tuning parameter and $\mathcal{R}_{seen}$ is the number of seen requests. The factor as a function of the number of seen requests is shown in Figure 6-3 for $\alpha = 80$.

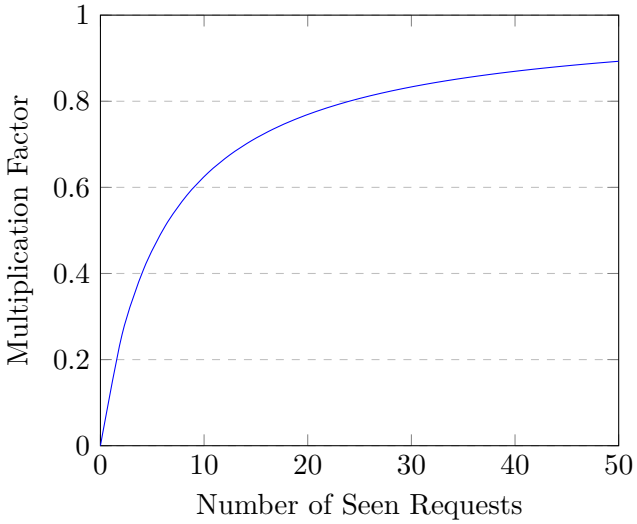


Figure 6-3: Expected pick-up multiplier as a function of the number of seen requests.

Algorithm 3 Future pick-up estimation for a vehicle $v \in \mathcal{V}$ and a feasible schedule $S \in \mathcal{S}$.

$g_v \leftarrow \text{GetRegion}(v)$
$d_0 \leftarrow \text{GetRateEstimate}(g_v)$
$p_0 \leftarrow \text{GetNumberOfPassengers}(v)$
$t_0 \leftarrow 0$
$t_f \leftarrow 15$ minutes
$l \leftarrow \text{Length}(S)$
for $i = 0$ **to** $i < l$ **do**
  $g_i \leftarrow \text{GetRegion}(S[i])$
  $d_{i+1} \leftarrow \text{GetRateEstimate}(g_i)$
  $t_{i+1} \leftarrow \text{GetEventTime}(S[i])$
  if $S[i] =$ pick-up **then**
    $p_{i+1} \leftarrow p_i + 1$
  else
    $p_{i+1} \leftarrow p_i - 1$
  end if
end for
$\pi \leftarrow 0$
for $i = 0$ **to** $i \leq l$ **do**
  if $t_{i+1} \leq t_f$ **then**
    $\mathcal{R}_{seen} \leftarrow \frac{1}{2}(d_{i+1} + d_i)(t_{i+1} - t_i)$
    $\pi \leftarrow \pi + \rho_s(\mathcal{R}_{seen}) \times \rho_l(p_i) \times (\text{GetCapacity}(v) - p_i)$
  else if $t_{i+1} > t_f$ **and** $t_i \leq t_f$ **then**
    $\hat{d} \leftarrow \text{LinearInterpolation}(d_{i+1}, d_i, t_{i+1}, t_i, t_f)$
    $\mathcal{R}_{seen} \leftarrow \frac{1}{2}(\hat{d} + d_i)(t_f - t_i)$
    $\pi \leftarrow \pi + \rho_s(\mathcal{R}_{seen}) \times \rho_l(p_i) \times (\text{GetCapacity}(v) - p_i)$
  end if
end for
if $\text{GetEventTime}(S[l-1]) < t_f$ **then**
  $\mathcal{R}_{seen} \leftarrow d_l \times (t_f - t_l)$
  $\pi \leftarrow \pi + \rho_s(\mathcal{R}_{seen}) \times \text{GetCapacity}(v)$
end if

Passengers On Board

The more passengers a vehicle has on board, the harder it gets to pick up additional passengers. This is because passengers on board might have already experienced a long waiting time, making the route very inflexible, or they might have destinations that are far apart, also making the schedule very rigid. It is difficult to determine exactly what the effect is on the number of additional passengers a vehicle can pick up, but a simple model can be constructed that roughly approximates the relation. A multiplication factor based on the number of passengers on board is defined as:

$$\rho_l = e^{-\frac{l^2}{\beta}} \tag{6-3}$$

In which l is the number of passengers on board of the vehicle, and β is a tuning parameter.

The load factor ρ_l is not vehicle specific, and should generalise for any vehicle capacity. For zero passengers on board, this factor is 1, and for infinite passengers on board it approaches 0. This expresses that for infinite passengers on board, the vehicle schedule is already so rigid that it becomes impossible to add any additional passengers to the vehicle while still respecting the time constraints of all passengers. The relation is shown in Figure 6-4 for $\beta = 80$.

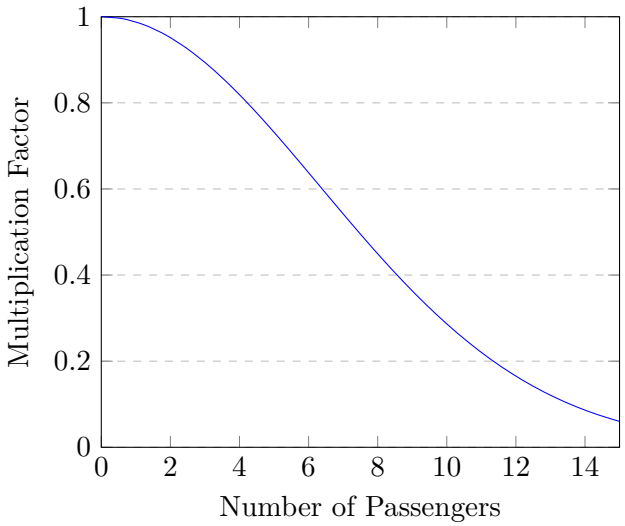


Figure 6-4: Expected pick-up multiplier as a function of the number of passengers on board.

Feed-back

In the proposed expected pick-up model, the individual positioning of vehicles is taken into account with respect to demand. It does not take into account how many other vehicles are also in a specific region. This could have as effect the oversupply of vehicles in specific areas of high demand, while too few vehicles are available in areas of relatively low demand. The number of vehicles available in a region has an effect on the number of pick-ups a vehicle can expect. For a given demand, the higher the number of vehicles in a certain region, the lower the chance that a request will end up in one particular vehicle. To avoid this, the expected number of pick-ups of a particular vehicle should also depend on the number of other vehicles in a region.

It is hard to model this effect in the cost for schedules. More importantly, it also has a profound impact on the way this problem can be solved. The schedule costs now depend on the vehicle distribution. Since vehicle distribution depends on the vehicle-schedule assignment, and the vehicle-schedule assignment in turn depends on the schedule costs, this problem cannot be solved using ILP methods. Although there are methods available to solve such non-linear integer programming problems, these are generally much slower.

There are several options to simplify this problem into a linear problem. A simple way of getting information of areas with an oversupply of vehicles, is to check the number of walk-away requests per region. The assumption is that in areas where requests are not served, there are too few vehicles available. This is implemented by keeping a list of requests that were

ignored in the previous 5 minutes. This list is updated before every schedule assignment. For every request in this list, it is determined in which region the origin of the request lies. The original estimated demand for the region is subsequently artificially increased linearly to the number of missed requests in that region. Since the estimated number of pick-ups depends on the demand in a region, this action will favour routes through these regions.

6-2-2 Cost Function

The vehicle-schedule assignment optimisation is now a multi-objective problem that should take into account both the cost of schedules according to the information currently available to the system and according to a predicted uncertain future state of the system. A new cost function can be formulated that expresses the trade-off between current schedule cost and the positioning for future requests as follows:

$$c_{total} = c_{schedule} - \sigma \cdot \pi \tag{6-4}$$

Where $c_{schedule}$ is the current schedule cost (sum of requests delays in previous chapters) and π is the predicted number of pick-ups over a given horizon according to the schedule. Furthermore σ is a weighing factor that determines the influence of the predicted requests relative to the influence of the current schedule cost. The trade-off expressed with σ is how much higher current schedule cost is acceptable if such a schedule is likely to yield future pick-ups. For every predicted future pick-up during the schedule, σ is subtracted from the schedule cost.

Time To Drop-Off

In all of the previous experiments, schedules are assigned to vehicles so that the sum of the delays of all requests are minimised. This cost function yields a relatively high service rate since this ensures that requests clear the system quickly and more seats are available sooner for future requests. Other attempts, such as penalising empty vehicle seats, or minimising schedule lengths has a detrimental effect on service rate. Doing so will pick up as many people as possible at the current time, but future incoming requests in that case cannot be serviced very well. Minimising delay in effect minimises the time requests spend in the system. A better cost function however might be to instead minimise the sum of request time to drop-offs (TTDs). The TTD is defined as the time between the schedule assignment and the time at which the vehicle is projected to reach the request's destination. In that case, schedules are assigned to vehicles so that requests are served as quickly as possible and seats become vacant as early as possible. The cost function is very similar to the delay cost function, however it does not care about the individual requests incurred delays (as long as it stays within the maximum specified delay constraint). Furthermore, whenever not all requests can be served, this cost function prefers to serve requests with a short travel time between their origin and destination. This cost function should yield better performance than the delay cost function.

6-3 Results

Several experiments were performed using different versions of the predicted future pick-up model. This was done to tune the model parameters and to determine the effect on different fleets. This led to the use of the tuning parameters $\sigma = 25$ seconds, $\alpha = 80$ and $\beta = 80$. Detailed results of several experiments are shown in Table 6-1 for the time to drop-off with predicted pick-ups (TTDPP) cost function. A comparison of the performance of vehicle-schedule assignment with use of the TTDPP cost function to the original delay cost function with both informed and naive rebalancing is presented in Figure 6-5 and Figure 6-6. These two rebalancing methods are described in Chapter 4. The method using delay in combination with naive rebalancing (D-NR) corresponds to the current state-of-the-art as presented in [2]. Figure 6-5 furthermore shows results for the TTD in combination with informed rebalancing (TTD-IR).

It can be observed that the TTDPP in combination with informed rebalancing (TTDPP-IR) outperforms all other implementations in all fleet configurations in terms of service rate. For a fleet of 1000 buses, the service rate is 2.62% higher than the service rate obtained with the D-NR, and 1.76% higher than delay in combination with informed rebalancing (D-IR). The second highest service rate for all fleet compositions is obtained with the TTD-IR, followed closely by the TTDPP in combination with naive rebalancing (TTDPP-NR). The TTD-IR performs comparable but slightly worse in terms of service rate to the TTDPP-IR for a fleet of only regular sized vehicles, but the TTDPP-IR outperforms the TTD-IR more significantly as more buses are introduced to the fleet.

From Figure 6-6, it can be observed that for all experiments, the TTDPP-IR yields the lowest in-car delay. The average in-car delay for a fleet of 1000 buses is approximately 45 seconds (14%) lower than with the D-NR and 23 seconds (8%) lower than with the D-IR. It can furthermore be seen that for the TTDPP with both the naive and informed rebalancer, the in-car delay decreases as the number of buses in the fleet is increased. The average waiting time varies only slightly between different methods. For a fleet of 1000 buses the average waiting time with the TTDPP-IR is 2% higher than with the D-IR but 2% lower than with the D-NR. The number of ignored requests, or walk-aways, for all fleet compositions is lowest with the TTDPP-IR and highest with the D-NR. The number of walk-aways with the TTDPP-IR is 15% lower than with the D-NR for a fleet of 1000 buses. Finally, the average distance travelled per vehicle during the 24 hour simulation varies strongly, and is around 100 kilometres higher for all experiments with the informed rebalancer compared to the naive rebalancer regardless of the vehicle-schedule assignment method.

Figure 6-7 shows the number of vehicles with more than 4 passengers on board over the whole experiment with the D-IR and the TTDPP-IR for a fleet of 1000 buses. It can be observed that the number of vehicles with more than 4 passengers on board is noticeably less with the TTDPP-IR between the 8:00 and 17:00 and is approximately the same during the other time in the experiment. The average occupancy of the vehicles in use is shown in Figure 6-8. A slight but noticeably lower average occupancy can be observed in many parts of the experiments with the TTDPP-IR.

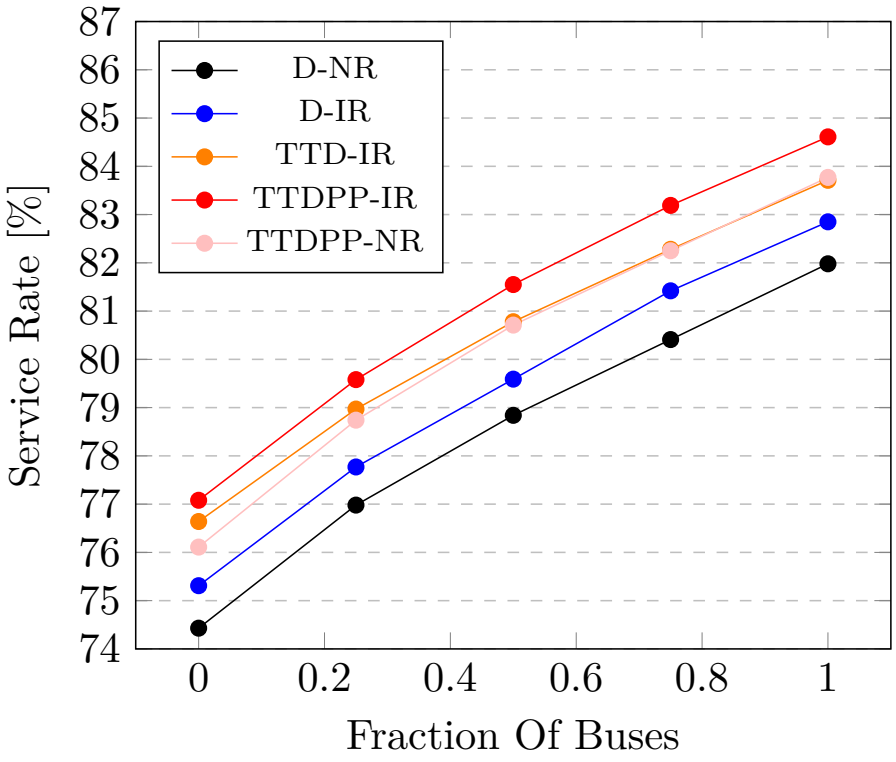


Figure 6-5: Comparison of service rate with different vehicle-schedule assignment methods in combination with different rebalancing methods. Service rates are shown for a heterogeneous fleet of 1000 vehicles with varying ratios of regular vehicles (capacity 4) and buses (capacity 8).

6-4 Discussion

From the experiments it is clear that for most performance metrics, the newly implemented TTDPP vehicle-schedule assignment method outperforms other strategies regardless of the rebalancing strategy used. For all fleet compositions, the experiments conducted with the TTDPP-IR yield the highest service rate and lowest in-car delay. The average distance travelled per vehicle for this method is high, but this is due to the rebalancing strategy. Similar improvements in service rate and in-car delay are obtained when comparing TTDPP-NR and D-NR, but without a significant increase in distance travelled. The average waiting time is marginally lower in comparison to the D-NR and marginally higher with the TTDPP-IR in comparison to the D-IR. The increase in average waiting time of 2% for a fleet of 1000 buses compared to the D-IR weighs up to the significant decrease in walk-aways and in-car delay.

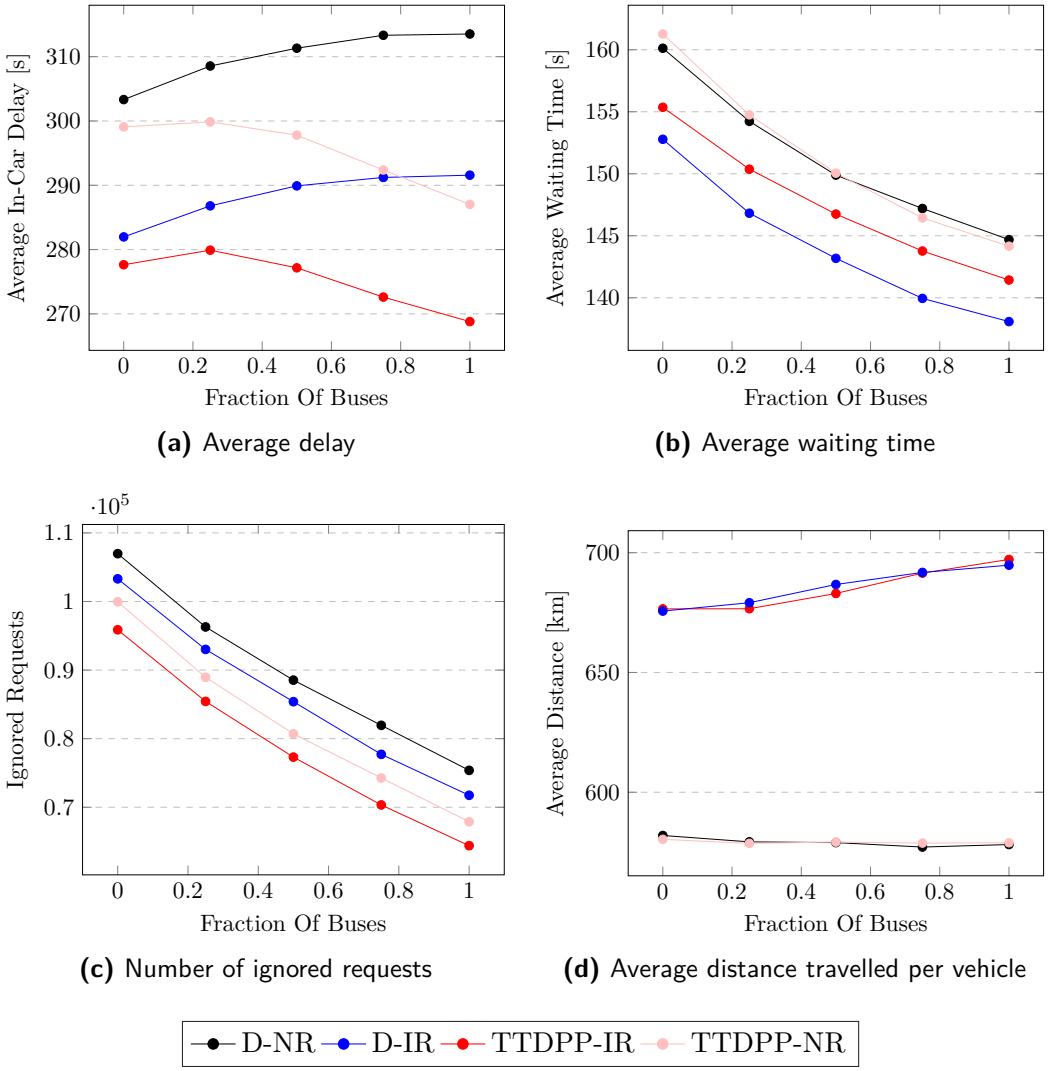
An interesting result is that in the experiment with the TTDPP-IR with 1000 buses the average occupancy and number of vehicles with more than 4 passengers on board during most of the experiment is slightly lower in comparison to the D-IR. It seems therefore that employing the TTDPP cost function makes the utilisation of high capacity vehicles worse. This is unexpected since the TTDPP strategy should bias vehicle routes through areas of high demand. In that case it seems that it is more likely that a vehicle can pick up more requests at the same time, and that is the reason the service rate is better. Alternatively the increase in service rate while having a decrease in average occupancy could be explained by

Vehicles of Capacity 8	Average In-Car Delay [s]	Average Waiting Time [s]	Average Distance [km]	Number Ignored	Serviced Requests [%]
0	277.64	155.36	676.54	95874	77.08
250	279.91	150.37	676.60	85435	79.58
500	277.16	146.75	682.98	77313	81.52
750	272.61	143.77	691.48	70337	83.19
1000	268.80	141.43	697.18	64402	84.61

Table 6-1: A detailed overview of the performance metrics for a heterogeneous fleet of 1000 vehicles with capacity 4 and 8 in varying ratios with $\Omega = 5$ minutes and $\Delta = 10$ minutes. These results are obtained using the TTDPP-IR.

the fact that the vehicles are more likely to be near future requests, because of which they can serve them faster and clear them from the system faster. This cannot be concluded however since the average waiting time varies only slightly with different implementations, and is even consistently slightly higher with the TTDPP-IR than with the D-IR method. However, the in-car delay obtained with the TTDPP-IR is significantly lower than with the D-IR, which means requests clear the system faster and explains the lower vehicle occupancy. The lower in-car delay is not due to the use of the TTD cost function instead of the delay cost function, since the in-car delay obtained for all fleet configurations using the TTD-IR and D-IR is almost the same. It seems that the effect is caused by the fact that the TTDPP-IR causes more vehicles to be present in areas of high demand. The requests in these regions can be served by more vehicles, which yields a larger number of feasible vehicle routes to choose from. This allows to select routes with lower TTDs, which explains a lower in-car delay. This also means that at any time in the simulation, more vehicle seats are vacant and more requests can be served.

The improvements in performance metrics are not spectacular, but are significant enough to demonstrate that the proposed vehicle-schedule assignment approach has a positive effect on fleet performance. This is the case especially for fleets that are barely big enough to serve demand, and in which there are few vehicles available for rebalancing (such as for a fleet of 1000 vehicles with the Manhattan taxi dataset). The future pick-up estimation model presented in this chapter is a very simple and inaccurate model that expresses to some extend which schedules are favourable for serving future requests. It is expected that more complex models, possibly trained using machine learning tools could improve the benefit of this approach further.



(a) Average delay

(b) Average waiting time

(c) Number of ignored requests

(d) Average distance travelled per vehicle

Figure 6-6: Comparison of several fleet performance metrics of a fleet of 1000 vehicles with varying ratios of vehicles of regular vehicles (capacity 4) and buses (capacity 8) for different vehicle-schedule assignment methods in combination with different rebalancing methods.

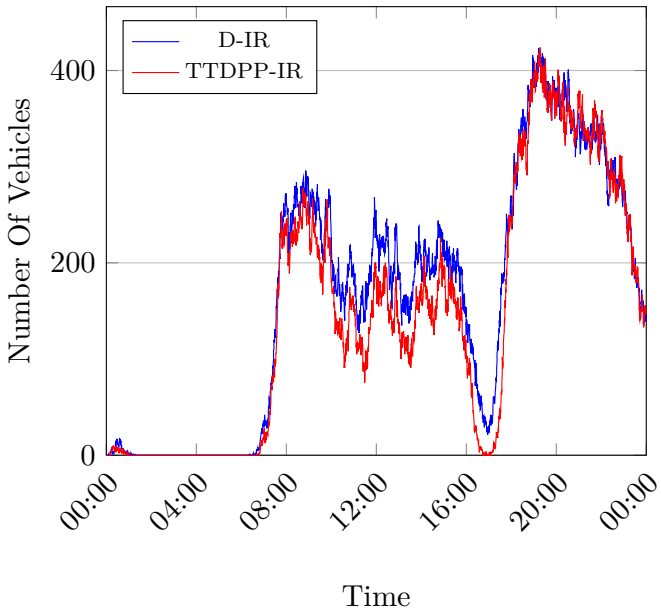


Figure 6-7: Number of vehicles with more than 4 passengers on board for a fleet of 1000 buses of capacity 8 for an experiment of 24 hours starting at midnight with $\Omega = 5$ minutes and $\Delta = 10$ minutes with the D-IR, and with the TTDPP-IR.

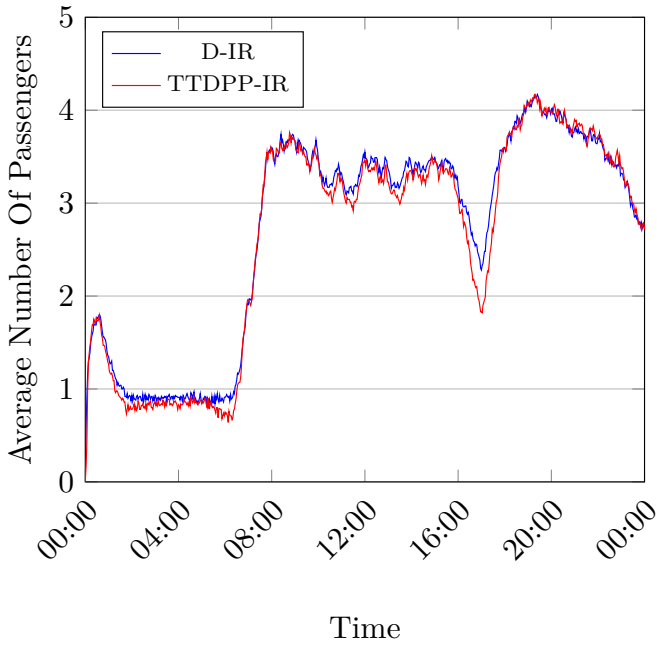


Figure 6-8: Average number of passengers on board for a fleet of 1000 buses of capacity 8 for an experiment of 24 hours starting at midnight with $\Omega = 5$ minutes and $\Delta = 10$ minutes with the D-IR, and with the TTDPP-IR

Chapter 7

Conclusion

The objective of this work was to develop an enhanced algorithm to dynamically route vehicles in a large scale Mobility-on-Demand (MoD) system in which multiple passengers can share the same vehicle. The focus was on improving the fleet performance in comparison to prior work in an urban environment with high densities of transportation requests and a fleet with a limited number of vehicles. For this work, the state-of-the-art in this field was reimplemented and extended with several features to improve fleet performance. Using these new features, significant improvements in fleet performance were obtained in terms of service rate, average delay and average waiting time.

Firstly a new rebalancing algorithm was presented in Chapter 4. A significant improvement in fleet performance was obtained by continuously rebalancing idle vehicles towards areas of expected demand. For a fleet of 3000 vehicles, a reduction in average waiting time of 37%, a reduction in travel time of 86% and a reduction in walk-aways of 95% was observed in comparison to the state-of-the-art. Furthermore it was shown that a fleet of 2000 vehicles with the new rebalancing algorithm could serve more requests within the service time constraints than a fleet of 3000 vehicles using the rebalancer used in the former state-of-the-art. This comes however at the cost of a significantly higher average travel distance by the vehicles.

Secondly the effect of vehicle capacities in homogeneous and heterogeneous fleets was analysed in Chapter 5. This work looked into the effect of increasing capacities in a homogeneous fleet, and the effect of introducing high-capacity buses to a fleet of regular sized vehicles in varying ratios. The experiments with a homogeneous fleet showed that for a given maximum waiting time and delay, the effect on service rate by increasing vehicle capacity is bounded. It was shown that for a fleet size of 1000 vehicles, a maximum waiting time of 5 minutes and a maximum delay of 10 minutes, the service rate increases steeply by increasing vehicle capacities from capacity 1 to 4. However, it does not improve significantly for capacities higher than 8. In experiments with a heterogeneous fleet of regular vehicles of capacity 4 and buses of capacity 8 in varying ratios, it was observed that with the current state-of-the-art vehicles of different capacities are not effectively utilised. In the vehicle-schedule assignment, vehicle capacities are not taken into account. A low number of buses is serving schedules that

could not have been served by regular vehicles, because of which the increase in performance by introducing buses to a fleet is relatively small.

Finally, motivated by the lack of performance increase by introducing buses to a fleet, this work looked into the effect of evaluating and selecting vehicle routes in anticipation to future requests which also takes into account the vehicle capacities. This was presented in Chapter 6. By favouring routes with high numbers of expected future pick-ups, vehicles of high capacities are more effectively utilised. Although the results showed that this caused a lower occupancy of high capacity vehicles, and an overall lower occupancy of all vehicles in the fleet, the number of serviced requests was increased. Compared to the state-of-the-art, for a fleet of 1000 vehicles the number of walk-aways was reduced by 15%. Furthermore a decrease in in-car delay of 14% was observed, with a marginal decrease of waiting time of 2%. This method in combination with the rebalancer from prior work furthermore outperforms the results presented for the new rebalancer for small fleets in terms of service rate and delay, without a significant increase in distance travelled. This new method is especially useful for small fleets of vehicles, where rebalancing has little effect due to the small number of idle vehicles.

7-1 Recommendations

The results presented in this work have demonstrated that significant theoretical fleet performance improvements are possible in a large scale MoD system by enhanced route optimisation algorithms. These improvements could contribute to the feasibility of such MoD systems as an alternative to privately owned cars. These contributions help make such a system more sustainable by decreasing the required number of vehicles, as well as more efficient from the perspective of both the user and the service provider. There are however several shortcomings and remaining opportunities in the methods presented in this work that should be addressed in future work before these methods can be applied in practical applications.

Firstly the scope of the dataset. The experiments were conducted with datasets of Manhattan only, for a limited time (Wednesdays for 24 hours) and with a limited number of different fleet configurations. Undoubtedly the results would have been different for different cities and for different days of the week. There are several reasons for the limited number of experiments conducted. Firstly, the Manhattan dataset is conveniently available, and is used in previous experiments as well, making it easier to compare results. The limited number of experiments and duration per experiment is due to the fact that running simulations takes considerable time. An average 24 hour simulation takes approximately 14 hours on a high-end consumer computer. It is advisable to test on different cities, for longer extends of time and with more fleet configurations to see how the results presented in this work generalise.

Secondly, a major shortcoming is that it is assumed in this work that there is no time required for passengers to be picked-up and dropped-off. This is unrealistic, and could make the results obtained better than what is possible in reality. Future work should take into account that vehicles need to stop and wait for passengers to enter and leave the vehicle.

Thirdly, the time to drop-off with predicted pick-ups (TTDPP) cost function that was presented in this work employed a very simple model to evaluate schedules in anticipation to future requests. This work has demonstrated that at least some improvement is possible

by introducing even a simple model that expresses which schedules are desirable for serving future requests. Further research should focus on the design of a more accurate model, possibly by using tools from machine learning. It is expected that further improvements in fleet performance could be obtained using such an improved model.

Furthermore, the experiments presented in this work have only considered a fleet with a fixed size and composition. It could also be possible to park vehicles that are not required at a depot and adapt the fleet to the demand at each time of day. In this way, it can be possible to utilise only the minimum required number of vehicles at any time to reach a certain minimum service performance.

Finally, it was chosen in this work to focus on increasing the number of served passengers with a given fleet. Other research should focus on the environmental and economic impact of routes and fleet configurations, perhaps using a multi-objective cost function that allows to trade-off between mileage, number of passengers served and number of required vehicles. In particular the rebalancer proposed in this work was not optimised to minimise mileage.

Appendix A

Alternative Schedule Function

This appendix shows the formulation of the schedule function as an integer linear programming (ILP). The schedule function computes for a vehicle and a set of requests the schedule with the lowest sum of delays to serve the requests. Let the following, known constants be defined:

- Ω is the maximum allowed waiting time.
- Δ is the maximum allowed delay.
- n is the number of requests.
- m is the number of passengers.
- Q is the capacity of the vehicle.
- R a set of indices $\{1, \ldots, n\}$ for all requests $\mathcal{R}$.
- P a set of indices $\{n+1, \ldots, n+m+1\}$ for all passengers of the vehicle.
- V_p set of indices $\{i \mid i \in R\}$ for all pick-up nodes.
- V_d set of indices $\{n+i \mid i \in R \cup P\}$ for all drop-off nodes.
- v the index 0, which is the vehicle start node.
- $V = V_p \cup V_d \cup v$ the set of indices for all nodes to be in the route.
- $t_{i,j}$ the travel time of a direct route between two nodes $i, j \in V$.
- $t_{r,i}$ the time at which request $i \in R \cup P$ was placed.
- α_i the minimum pickup to drop-off node travel time of a request or passenger $i \in R \cup P$.
- $t_{p,i}$ the time that a passenger $i \in P$ was picked up.

- d_i the service time of a node i. This could be defined for all pick-up and drop-off nodes according to the average time it takes to pickup and drop-off passengers.

- q_i is the load at every node $i \in V$. It is defined by how many people are picked up or dropped off at every node. For all pickup nodes it is equal to the number of people in the associated request, and for drop-off locations it is equal to the negative of that. For the vehicle node it is equal to 0.

- t_{cur} is the current time (the time at the start of the optimisation).

- $[e_i, l_i]$ is the time window between which every node should be visited for $i \in V$. Here, e_i and l_i respectively are the earliest and latest time that node i can be visited. For all pick-up nodes $i \in V_p$ these are defined as $e_i = t_{r,i}$ and $l_i = t_{r,i} + \Omega$. For all drop-off nodes $i \in V_d$ these are $e_i = t_{r,i-n} + \alpha_{i-n}$ (no earlier than the time at which the request was placed plus the minimum travel time between pick-up and drop-off locations), and $l_i = e_i + \Delta$ (no later than the earliest possible drop-off time plus the maximum delay).

Furthermore let the following ILP variables be defined:

- $\mathcal{X} = \{x_{i,j} \mid i \in V, j \in V \setminus v, i \neq j, j \neq i - n\}$, in which $x_{i,j}$ is equal to 1 if the vehicle travels directly from node i to node j and 0 otherwise. This variable describes the schedule.

- $\mathcal{U} = \{u_i \mid i \in V\}$, in which u_i is the time at which each node is visited.

- $\mathcal{W} = \{w_i \mid i \in V\}$, in which w_i is the number of passengers on board of the vehicle when the vehicle leaves node i.

MILP Formulation

With the defined constants and variables, the ILP can be formulated as follows:

$$\min_{U} \sum_{i \in R} (u_{i+n} - (t_{r,i} + \alpha_i)) + \sum_{i \in P} (u_i - (t_{r,i} + \alpha_i)) \tag{A-1}$$

subject to:

$$\sum_{j \in V} x_{i,j} = 1 \qquad \forall i \in V_p \cup v \tag{A-2}$$

$$\sum_{i \in V} x_{i,j} + x_{j,i} \leq 1 \qquad \forall j \in V \tag{A-3}$$

$$\sum_{i \in V} x_{i,j} = 1 \qquad \forall j \in V_p \cup V_d \tag{A-4}$$

$$\sum_{j \in V} x_{i,j} \leq 1 \qquad \forall i \in V_d \tag{A-5}$$

$$u_j \geq u_i + d_i + t_{i,j} - M_{i,j}(1 - x_{i,j}) \qquad \forall i, j \in V \tag{A-6}$$

$$w_j \geq w_i + q_j - W_{i,j}(1 - x_{i,j}) \qquad \forall i, j \in V \tag{A-7}$$

$$w_v = m \tag{A-8}$$

$$u_v \geq t_{cur} \tag{A-9}$$

$$r_i \geq u_{i+n} - (u_i + d_i) \qquad \forall i \in R \tag{A-10}$$

$$r_i \geq u_i - t_{p,i} \qquad \forall i \in P \tag{A-11}$$

$$e_i \leq u_i \leq l_i \qquad \forall i \in V_p \cup V_d \tag{A-12}$$

$$\alpha_i \leq r_i \leq \Delta + \alpha_i \qquad \forall i \in R \cup P \tag{A-13}$$

$$u_{i+n} \geq u_i \qquad \forall i \in R \tag{A-14}$$

$$max(0, q_i) \leq w_i \leq min(Q, Q + q_i) \qquad \forall i \in V \tag{A-15}$$

$$x_{i,j} \in \{0, 1\} \tag{A-16}$$

In this formulation, the objective function is the sum of delays of the passengers and requests. Constraint A-2 ensures that from every pickup node and the vehicle node, there is exactly one route going to another node (a pickup node can never be the final node to be visited in a route). Constraint A-3 makes sure that a route between two nodes is never traversed twice. This avoids the occurrence of loops. Constraint A-4 ensures that there is an incoming route for all nodes, except the vehicle node (this is the start node of the route). Constraint A-5 ensure that at most 1 route leaves every destination node. This is at most because the last node in the route (which can only be a destination node) has no outgoing route. Constraint A-6 defines the service time u_i of every node. It makes sure that if a node j is visited later than node i, the time that node j is visited is always larger or equal to the sum of the time that node i was visited, the service time of node i and the travel time between node i and node j. $M_{i,j}$ is used as a linearisation parameter and is defined as $M_{i,j} = max(0, l_i + d_i + t_{i,j} - e_j)$ [14]. Constraint A-7 defines the load of the vehicle at every node. It ensures that if in a route, node j is visited directly after node i the load of a vehicle after visiting node j is greater or equal to the sum of the load of the vehicle at node i and the number of people picked up (or dropped off) at node i. Similar to constraint A-6, the variable $W_{i,j}$ is a linearisation parameter that is defined as $W_{i,j} = min(Q, Q + q_i)$. Constraint A-8 ensures that the initial load of the vehicle is equal to the number of passengers already on board. Constraint A-9 ensures that the time the route starts is greater or equal to the current time. Constraint A-10

defines the ride time of all requests. It ensures that this is larger or equal to the time the request drop-off location was visited minus the time that the pick-up location was visited, while taking into account the service time of the pickup node. Constraint A-11 defines this for a passenger, where the pickup time is already known at the start of the optimisation. Constraint A-12 ensures that all nodes are visited within the set time windows. Constraint A-13 ensures that the ride time of all requests and passengers is larger or equal to the direct pickup to destination travel time but smaller than the sum of the direct travel time and the maximum allowed delay. Constraint A-14 ensures that for every request, the destination node is always visited later than the origin node. Constraint A-15 ensures that the vehicle load never becomes negative or that the capacity of the vehicle is exceeded. Finally constraints A-16 limits the value of the binary variable $x_{i,j}$ to 0 and 1.

A-1 Performance

The performance of this new travel function was not as expected. Most likely due to the overhead of defining the problem in Gurobi. For a trip of size 1, the computational time was around 1 ms on an 2,3 GHz Intel Core i7. For trips of size 2 this is approximately 2 ms, size 3 is 5 ms and for size 4 is 26 ms. Especially the trip of size 1 with a computational time of 1 ms shows the inefficiency of using Gurobi. For a trip of size 1 there is only 1 possible route, and the only thing that has to be checked is if the maximum waiting time is met. This should take much shorter than 1 ms.

Appendix B

Rebalancing

Often the demand for vehicles is not uniformly distributed over an operating area. Furthermore, since rides are often booked from areas of high demand to areas of low demand, vehicles have a tendency to build up in areas with low demand. This mismatch between vehicle supply and demand has a negative effect on passenger waiting time and potentially even leads to requests that cannot be served within the maximum allowed time constraints. It is therefore important to proactively rebalance idle vehicles towards areas of high demand. This thesis is connected to a paper that is under review in which a method is proposed to estimate demand over the operating area, and a method to optimally assign idle vehicles to travel to areas in the operating area according to this estimated demand.

The paper was written in close cooperation in both writing and the development of the software used for experimentation. The same software and the algorithms described in the paper are also used in this thesis.

Vehicle Rebalancing for Mobility-on-Demand Systems with Ride-Sharing

Alex Wallar*, Menno van der Zee†, Javier Alonso-Mora† and Daniela Rus*

***Abstract*— Recent developments in Mobility-on-Demand (MoD) systems such as ride-sharing and vehicle-pooling have demonstrated the potential of road vehicles as an efficient mode of urban transportation. Newly developed algorithms can compute vehicle routes in real-time for batches of requests and allow for multiple requests to share vehicles. These algorithms have primarily focused on optimally producing vehicle schedules to pick up and drop off requests. The redistribution of idle vehicles to areas of high demand, known as *rebalancing*, on the contrary has received little attention in the context of ride-sharing. In this paper, we present a method to rebalance idle vehicles in a ride-sharing enabled MoD fleet. This method consists of an algorithm to optimally partition the fleet operating area into rebalancing regions, an algorithm to determine a real-time demand estimate for every region using incoming requests, and an algorithm to optimally assign idle vehicles to these rebalancing regions using an integer linear program. Evaluation with historical taxi data from Manhattan shows we can service 99.8% of taxi requests in Manhattan using 3000 vehicles with an average waiting time of 57.4 seconds and an average in-car delay of 13.7 seconds. Moreover, we can achieve a higher service rate using 2000 vehicles than prior work achieved with 3000. Furthermore, with a fleet of 3000 vehicles, we reduce the average travel delay by 86%, the average waiting time by 37%, and the amount of ignored requests by 95% compared to earlier work.**

I. Introduction

Autonomous vehicles are enabling a new era of personal mobility with the promise of transportation available anywhere, anytime. Scalable algorithms for motion planning and fleet management that can cope with large request loads are needed for urban deployments. Efficiencies provided by the notion ride-sharing will help optimize those services. In this paper we propose a scalable real-time algorithm for large scale management of fleets of vehicles (autonomous or human-driven), under the ride-sharing model. We build on our previous work [1] and study the role of rebalancing in optimizing a fleet management system.

Ride sharing services such as UberPool and Lyft Line have demonstrated the potential for road vehicles to be used as a sustainable and effective mode of passenger transportation in urban environments. The introduction of e-hailing and global vehicle dispatching in these Mobility-on-Demand (MoD) systems have opened up the opportunity to assign vehicles to requests more efficiently. By allowing multiple passengers to share a single vehicle and considering batches of requests placed around the same time, vehicle routes can be optimized so that less vehicles can serve more requests. This makes the MoD fleet more affordable, sustainable and time-effective, which will be further enhanced by the introduction of autonomous vehicles.

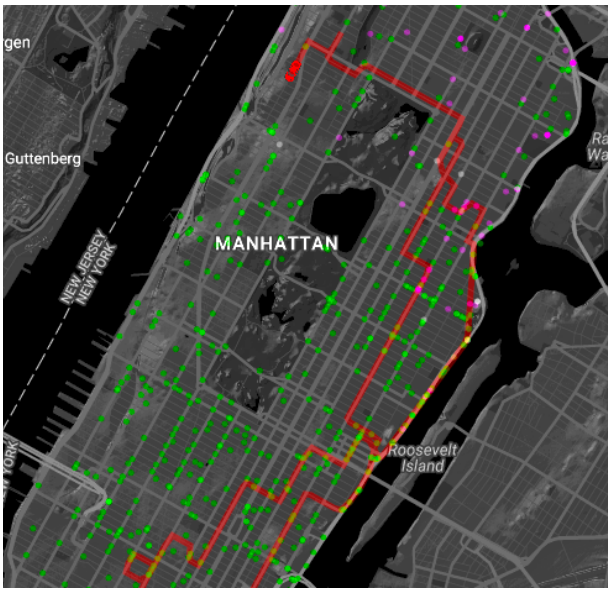


Fig. 1: A snapshot of the simulator. Green dots represent vehicles that either have passengers on board or are on their way to pick up passengers. Pink dots represent vehicles that are rebalancing. Grey dots represent vehicles that are idle. The snapshot furthermore shows the location and history of one of the vehicles in the fleet (represented by the red car).

Besides the computation of efficient vehicle schedules, the proactive relocation of idle vehicles can have a significant influence on the fleet performance. Since the demand for vehicles is often not uniformly distributed, vehicles tend to build up in regions of low demand while vehicles are depleted in regions of high demand. For example in Manhattan, there are many trips to Harlem at night, but fewer back to Midtown in the morning. This mismatch in vehicle supply and demand means that vehicles often have to travel further than necessary to pick up customers, which leads to higher waiting times and more customer walk aways. It also means that the number of passengers which a fleet can transport in a

* The authors are at the Computer Science and Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, 32 Vassar St, 02139 Cambridge MA, USA {wallar,rus}@csail.mit.edu

† The authors are at the Department of Cognitive Robotics, Delft University of Technology, Mekelweg 2, 2628 CD Delft, Netherlands M.J.vanderZee@student.tudelft.nl, J.AlonsoMora@tudelft.nl

*This work was supported in part by the Netherlands Organisation for Scientific Research NWO Veni 15961 and the MIT-Singapore Alliance on Research and Technology under the Future of Urban Mobility.

given time is less than optimal. Vehicle rebalancing focuses on positioning the idle vehicles so that future demand can be served with increased efficiency.

In this work, we build on the vehicle schedule optimization algorithm presented in [1] and expand it with a method to continuously rebalance idle vehicles. We present a method to determine an optimal discretization of the operating area into well defined rebalancing regions and a method to estimate, from incoming transportation requests, a real-time demand per region. Using this estimation we subsequently optimize the rebalancing of idle vehicles towards rebalancing regions. Finally, we present a case study using real taxi data from Manhattan to demonstrate the benefits of our rebalancer and compare it to the previous state of the art.

A. Related works

Several works have looked into the redistribution of idle vehicles in a fleet to meet future demand. Most have looked into the redistribution of vehicles in one-way car sharing schemes such as car2go and Zipcar, or bike-sharing schemes. Such systems experience similar mismatches between vehicle supply and demand. Many of these works however focus on infrequent rebalancing (in the order of several times a day) [2], the practical implications of use of human operators to rebalance vehicles [3], or on theoretical formulation and experimentation of the optimization problem [4]. Prior work has also studied how vehicles can be redistributed to fixed stations by drivers employed by a fleet manager [5], [6]. However these approaches do not directly apply to the continuous pick up and delivery paradigm of mobility-on-demand.

More relevant to this paper are rebalancing techniques for autonomous mobility-on-demand systems, a car sharing scheme in which vehicles are able to redistribute themselves without the intervention of a human operator. Much of this research shows a significantly reduced fleet size is required to serve a fixed demand with a similar quality of service [7], [8]. However, these works often use long rebalancing intervals [7] or use simplified models to simulate demand and vehicle movement [8].

A similar approach to the current work is presented in [9]. However, this approach used predicted demands learned from historical data, which requires elaborate historical data to be available, and presented a naive strategy for rebalancing idle vehicles which assigned idle vehicles to ignored requests.

B. Contribution

Working further on the earlier work presented in [1] that computes an optimal assignment of a fleet of autonomous or human-driven vehicles to a set of requests, we present a method that continuously rebalances the remaining idle vehicles over the operating area according to estimated real-time demands. Specifically, we present:

- A method to optimally discretize the operating area into a set of rebalancing regions.
- A method to estimate vehicle demand for every rebalancing region using only the real-time request stream.
- An algorithm to assign idle vehicles to rebalancing regions using the estimated demand.
- Experimental validation comparing the performance of using no rebalancer, the rebalancer presented in prior work, and the new proposed rebalancing strategy.

II. Preliminaries

In this section, all relevant notations used in this paper is presented. Furthermore, we define the problem and present a general overview of the methods employed.

A. Definitions

We consider a fleet of vehicles $\mathcal{V}$ which can either be autonomous or human-driven. Every vehicle $v \in \mathcal{V}$ is defined by the tuple $\{q_v, \mathcal{P}_v, \kappa_v\}$, in which q_v is the vehicle's current location, $\mathcal{P}_v = \{p_1, \ldots, p_n\}$ is a set of passengers, and κ_v is the vehicle's capacity. A passenger is defined as a request that has been picked up by a vehicle.

We consider set of transportation requests $\mathcal{R} = \{r_1, \ldots, r_n\}$ defined by the tuple $\{o_r, d_r, t_r^r, t_r^{pl}, t_r^*\}$. Here, o_r is the origin, d_r is the destination, t_r^r is the time that the request was placed, t_r^{pl} is the latest acceptable pickup time and t_r^* is the earliest possible time that the destination could be reached. Furthermore, the actual request pick-up time is denoted by t_r^p, and the drop-off time is denoted by t_r^d.

Vehicles move according to schedules that they are assigned. A schedule S is defined as a sequence of request pick-up and drop-off events, and describes in which order requests are picked up and dropped off.

We define an operating area comprised of a road network as the region the vehicles will consider requests. This operating area is partitioned into a set of rebalancing regions, $\mathcal{G}$, with region centres, $\mathcal{C}$ (described in Sec. III-A). All locations closer to a region centre $c \in \mathcal{C}$ than any other region centre in $\mathcal{C}$ belong to the associated region.

Additionally, let us define for every request a waiting time $\omega_r = t_r^p - t_r^r$ and a travel delay time $\delta_r = t_r^d - (t_r^r + t_r^*)$. The waiting time and travel delay represent the total amount of time a request waits to be picked up by a vehicle and the time added to the request's transit as opposed to directly travelling to their destination respectively. We also define maximum allowed wait time Ω and maximum allowed delay time Δ as in [1]. The values Ω and Δ are used as service metrics.

B. Problem Formulation

After every predetermined time interval ψ, schedules are computed and assigned to vehicles so that the sum of delays of all requests is minimized. This step will be referred to as the schedule assignment. For a detailed explanation of the algorithm used to assign vehicle schedules, we refer to [1]. In some cases, not all vehicles in the fleet are assigned a schedule. This is either because there are no requests within a travel time smaller than Ω or because there are other vehicles available that are able to serve the requests more efficiently. Let us denote this set of unassigned vehicles by $\mathcal{V}_r \subseteq \mathcal{V}$. These are the vehicles that are considered for rebalancing.

$\mathcal{G}$	Set of rebalancing regions
$\mathcal{C}$	Set of rebalancing region centers
T_{ij}	Travel time from vertex i to j in road-network
$\tilde{\lambda}_j$	Current estimate of rate of requests in region j
$\mathcal{V}_r$	Set of vehicles to rebalance
$\mathcal{H}$	Time horizon for rebalancing
ψ	Interval time for batch assignment
Ω	Maximum waiting time
Δ	Maximum travel delay

TABLE I: Notation used throughout the paper

In both cases, there is an oversupply of vehicles in those particular regions. At the same time, other regions might have an under supply of vehicles. In that case, requests in those regions might have to wait significantly longer before they are picked up and eventually might not be able to be serviced while respecting the constraints set by Ω and Δ.

The focus of this paper is to determine how to distribute the unassigned vehicles over the operating area such that the request delay and waiting time is reduced, the number of ignored requests is minimized, and to do this dynamically over time.

C. Method overview

The method to assign vehicle schedules and rebalance idle vehicles is split up into multiple steps. First, using an integer linear program (ILP), the operating area is discretized into the set of regions $\mathcal{G}$ with region centres $\mathcal{C}$. The regions are computed once offline, and remain constant during the online schedule assignment. This process is explained in Sec. III-A. A schematic overview of the steps performed at every assignment interval ψ is shown in Fig. 2. The following steps are performed:

1) Vehicles are assigned schedules to pick up and drop off requests using the algorithm presented in [1].
2) Using the real-time request information, the current demand at every rebalancing region is estimated using a particle filter. See Sec. III.
3) The vehicles that remained unassigned in the vehicle-schedule assignment (step 1) are assigned to rebalance towards regions in $\mathcal{G}$. This rebalancing assignment is computed using an ILP. See Sec. IV.

All vehicle schedules and rebalancing assignments are re-considered at every assignment interval. Previously assigned vehicle schedules can change at each subsequent schedule assignment, and vehicles on the way to a rebalancing region can instead be assigned a schedule to pick up passengers. A detailed description of the method overview can be found in Algo. 1.

III. Demand Estimation

During the execution of the algorithm, we estimate the rate at which incoming requests are being introduced at different locations in our operating area. We do so by first partitioning our operating area into a number of regions and for each region, utilizing a particle filter to estimate the demand.

Algorithm 1 Overview of the rebalancing method

$\mathcal{G} \leftarrow$ DiscretizeOperatingArea(t_{max})
for all $g \in \mathcal{G}$ **do**
  InitializeRateEstimate$_g$()
end for
for every time interval, ψ **do**
  $\mathcal{R} \leftarrow$ IncomingRequests()
  AssignVehicleSchedules($\mathcal{V}, \mathcal{R}$)
  $Q_g \leftarrow 0 \quad \forall g \in \mathcal{G}$
  for all $r \in \mathcal{R}$ **do**
    $g \leftarrow$ GetRebalancingRegion(r)
    $Q_g \leftarrow Q_g + 1$
  end for
  for all $g \in \mathcal{G}$ **do**
    UpdateRateEstimate$_g$(Q_g, ψ)
  end for
  $\mathcal{V}_r \leftarrow$ GetRebalancingVehicles()
  RebalanceVehiclesToRegions($\mathcal{V}_r, \mathcal{G}$)
end for

A. Discretization into regions

Given a directed graph, $G = (V, A)$, representing the road network where V is the set of vertices, and a matrix T where T_{ij} represents the travel time between vertex i and j, our problem is to select a subset of vertices $\mathcal{C} \subseteq V$ as region centers that can be used to aggregate demand. A request is in region $g \in \mathcal{G}$ if its origin is closer to the region center c of region g than any other region center in $\mathcal{C}$.

We discretize the operating area into regions using given parameter t_{max} which represents the maximum travel time between any vertex in the graph and the closest region center. To determine the minimum number of regions for a given t_{max}, we formulate the problem as an ILP.

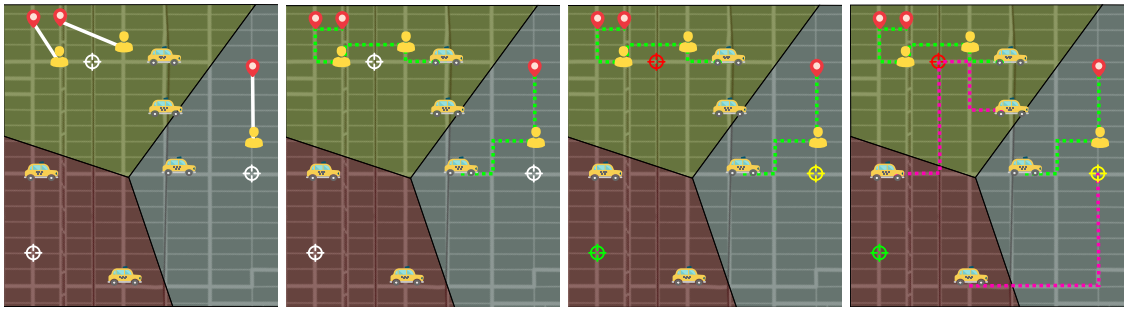
First we define a reachability matrix, R, where $R_{ij} = 1$ if $T_{ij} \leq t_{max}$ and $R_{ij} = 0$ otherwise. This describes whether vertex j is reachable from vertex i given the time limit. We also define a set of binary variables x where $x_i = 1$ if vertex V_i is used as a region center and 0 otherwise. Using the reachability matrix and the binary variables, we can define an ILP to determine the minimum number of region centers such that every vertex in the graph is reachable from at least one region center as:

$$\min_{x} \quad \sum_{i=1}^{|V|} x_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{|V|} x_i \cdot R_{ij} \geq 1 \quad \forall j \in [1, |V|] \tag{2}$$

Eq. (2) ensures that every node in the road network graph is reachable within t_{max} travel time by at least one region center selected from the nodes in the graph. To extract the region centers, we select from V all vertices V_i such that $x_i = 1$.

The region centers are computed a priori and are used to aggregate requests together so the rate of requests for each



(a) Initial state (b) Schedule assignment (c) Demand estimation (d) Rebalancer assignment

Fig. 2: Schematic overview of the method used for the assignment of vehicles to requests and the rebalancing of vehicles in the order as they are performed. (a) Example for a part of a road network with three regions (white marker = region center), 5 vehicles, and three requests (yellow human = origin, red marker = destination). (b) Assignment of schedules to vehicles, the schedule trajectories are shown by the green dotted lines. Three of the five vehicles are not assigned a schedule. (c) The estimation of demands for every region according to the request information. In this figure, high demand is represented by the red marker, intermediate demand by the yellow marker and low demand by the green marker. (d) Optimized assignment of unassigned vehicles to rebalancing regions. The rebalancing vehicles move towards the region centers. The rebalancing trajectories are shown in the purple dotted lines.

region can be computed. These region centers are also used for rebalancing as they are the locations that vehicles are proactively sent to.

B. Determining the rate of requests

We estimate the vehicle demand online in each rebalancing region using only the real-time request stream. We define the vehicle demand as the rate at which requests are originating from a given region over time.

The rate of requests at each region $g \in \mathcal{G}$ is modelled as an inhomogeneous Poisson point process with a stochastic time-varying rate, $\lambda_g(t)$. These rates are assumed to drift over a short time horizon according to a Wiener process. This means that for each region g, the change in rate of requests over time follows a Gaussian distribution, i.e. $\lambda_g(t') - \lambda_g(t) \sim \mathcal{N}(0, \nu \cdot (t'-t))$ for $t' > t$ and some given volatility parameter, ν. The rate, $\lambda_g(t)$, for each region is estimated using a sequential importance resampling particle filter as described in [10]. particle filter is updated with the number of requests observed, n, within a time interval, $t-\epsilon_t$ to t. The N particles, $\{\hat{\lambda}_0^{(i)} : 1 \leq i \leq N\}$, are initialized uniformly at random within an given upper and lower bound at time 0. Their weights, $\{w_0^{(i)} : 1 \leq i \leq N\}$, are all set to $1/N$. The particles are updated in four steps.

1) N samples, $\{\hat{\lambda}_{t-\epsilon_t}^{(i)}\}$ with weights, $\{w_{t-\epsilon_t}^{(i)}\}$, are drawn with replacement from the particles with probabilities proportional to their weights.
2) Random noise is applied to each particle according to the Wiener process: $\hat{\lambda}_t^{(i)} = \hat{\lambda}_{t-\epsilon_t}^{(i)} + \epsilon_\lambda$, where $\epsilon_\lambda \sim \mathcal{N}(0, \nu \cdot \epsilon_t)$
3) The weights are updated with the observation of n requests in ϵ_t time: $\tilde{w}_t^{(i)} = w_{t-\epsilon_t}^{(i)} \cdot \Pr[k = n; \epsilon_t \cdot \hat{\lambda}_t^{(i)}]$, where $\Pr[k = n; \epsilon_t \cdot \hat{\lambda}_t^{(i)}]$ is the Poisson probability of n events with a rate $\epsilon_t \cdot \hat{\lambda}_t^{(i)}$
4) The weights are normalized: $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_k \tilde{w}_t^{(k)}}$

The estimate of the stochastic rate, $\lambda_g(t)$, for a region g at time t is then defined as the weighted average of the particles, $\lambda_g(t) = \sum_i w_t^{(i)} \cdot \hat{\lambda}_t^{(i)}$. The particle filter produces an estimate of the rate of requests for a given region by estimating the likelihood of a fixed number of candidate rates and returning the likelihood weighted average over the candidate rates.

IV. Rebalancing

Due to the fact that demand is not equally distributed over the operating area, vehicles will tend to build up according to a spatial distribution that does not match the distribution of demand. Due to this undesirable distribution of vehicles, it is possible that some vehicles remain idle while there are requests that are not served. This takes places when there are no vehicles that can reach these requests within the maximum waiting time Ω, or when the demand in a specific region is very high and there are not enough vehicles to serve all the requests in that specific region. In order to mitigate this problem, idle vehicles should be proactively rebalanced over the operation area so that their distribution matches the distribution of demand. This furthermore decreases waiting time since for incoming requests, the probability of having a nearby vehicle available is higher. We propose a novel vehicle rebalancer that models the problem as an ILP to match the supply of vehicles to each area with the demand.

A. Implementation

Our rebalancer seeks to match the supply of idle vehicles in each region to the expected demand for a given time horizon $\mathcal{H}$. Let us define, $\mathcal{V}_r \subseteq \mathcal{V}$ as the set of idle vehicles that are available for rebalancing. These are the vehicles that are not assigned to pick up or drop off requests in the vehicle schedule assignment. Let us also define $\mathcal{C}$ as the set of region

centres as described in Sec. III-A. Our goal is to find an assignment from vehicles in $\mathcal{V}_r$ to region centres in $\mathcal{C}$ such that we maximise the amount of requests the vehicles are able to serve, while not over saturating or under saturating regions with vehicles.

To solve this assignment, we can formulate the problem as an ILP. Let us first define a set of binary variables, $\mathcal{X} = \{x_{ij} : \forall i \in [1, |\mathcal{V}_r|], \forall j \in [1, |\mathcal{C}|]\}$, where $x_{ij} = 1$ if vehicle i is assigned to rebalance to region centre j and zero otherwise. Let us also define a travel time matrix, T, where T_{ij} is the travel time from vehicle i to the region centre j and a rate vector $\tilde{\lambda}$ where $\tilde{\lambda}_i$ is the current rate of requests at region i computed using the particle filter described in Sec. III-B. With these variables, we can define the objective function for our ILP which we seek to maximize as:

$$\mathcal{J}(\mathcal{X}) = \sum_{i=1}^{|\mathcal{V}_r|} \sum_{j=1}^{|\mathcal{C}|} x_{ij} \cdot \tilde{\lambda}_j \cdot (\mathcal{H} - T_{ij}) \tag{3}$$

This objective represents the sum of the expected number of requests each vehicle would observe in its assigned rebalancing region for the given time horizon, $\mathcal{H}$. The expected number of requests observed by vehicle i is expressed as the rate of requests at the assigned region, $\tilde{\lambda}_j$, multiplied by the time remaining in the time horizon after the vehicle reaches the region, $\mathcal{H} - T_{ij}$

A valid rebalancing assignment must guarantee that each vehicle is assigned to at most one station. This is described in the constraint:

$$\sum_{j=1}^{|\mathcal{C}|} x_{ij} \leq 1 \quad \forall i \in [1, |\mathcal{V}_r|] \tag{4}$$

Also, due to our formulation, we constrain the solution to assign vehicles to rebalancing regions that are reachable within the time horizon, $\mathcal{H}$, i.e. $\mathcal{H} \geq T_{ij}$. This constraint is formulated as:

$$\begin{gathered} x_{ij} \cdot (\mathcal{H} - T_{ij}) \geq 0 \\ \forall i \in [1, |\mathcal{V}_r|] \text{ and } j \in [1, |\mathcal{C}|] \end{gathered} \tag{5}$$

In order to obtain an adequate dispersion of vehicles and limit the oversaturation of vehicles in rebalancing regions, we need to constrain the assignment such that the supply of vehicles in a rebalancing region is less than some factor of their demand. The supply of vehicles in region $j \in [1, |\mathcal{C}|]$ for a given time horizon can be written as:

$$\sum_{i=1}^{|\mathcal{V}_r|} x_{ij} \cdot \frac{\mathcal{H} - T_{ij}}{\mathcal{H}} \tag{6}$$

The supply of vehicles is weighted by the percent of time in the next time horizon a vehicle would be able to sit idle at the assigned station. The time weighting is used to give a more accurate estimation of the vehicle supply. For example, if a vehicle takes 8 minutes to reach a region and the time horizon is set to 10 minutes, that vehicle's supply is only available for 20% of the time.

Fig. 3: The computed location of region centers in Manhattan using the algorithm presented in Sec. III for a maximum reachability time $t_{max} = 150$ seconds.

The demand for vehicles for some region $j \in [1, |\mathcal{C}|]$ and a given time horizon is defined as:

$$\tilde{\lambda}_j \cdot \mathcal{H} \tag{7}$$

Putting Eq. (6) and (7) together we formulate a constraint to limit the oversaturation of vehicles in rebalancing regions as:

$$\sum_{i=1}^{|\mathcal{V}_r|} x_{ij} \cdot (\mathcal{H} - T_{ij}) \leq \tilde{\lambda}_j \cdot \mathcal{H}^2 \cdot \rho \quad \forall j \in [1, |\mathcal{C}|] \tag{8}$$

Note that for a more concise description, the time horizon, $\mathcal{H}$, was multiplied on both sides of the inequality. Also note that we have introduced a tuning parameter, ρ, that allows us to specify an acceptable level of oversaturation at a rebalancing region.

Combining the objective function from Eq. (3), $\mathcal{J}(\mathcal{X})$, with the constraints described in this section, we formulate an ILP that finds an assignment of vehicles to rebalancing regions that maximizes the expected number of requests observed by all vehicles while obtaining an adequate dispersion of vehicles to limit the oversaturation of vehicles in rebalancing regions. This ILP is then:

$$\begin{aligned} \max_{\mathcal{X}} \quad & \mathcal{J}(\mathcal{X}) \\ \text{s.t.} \quad & \text{constraints } (4), (5), (8) \end{aligned} \tag{9}$$

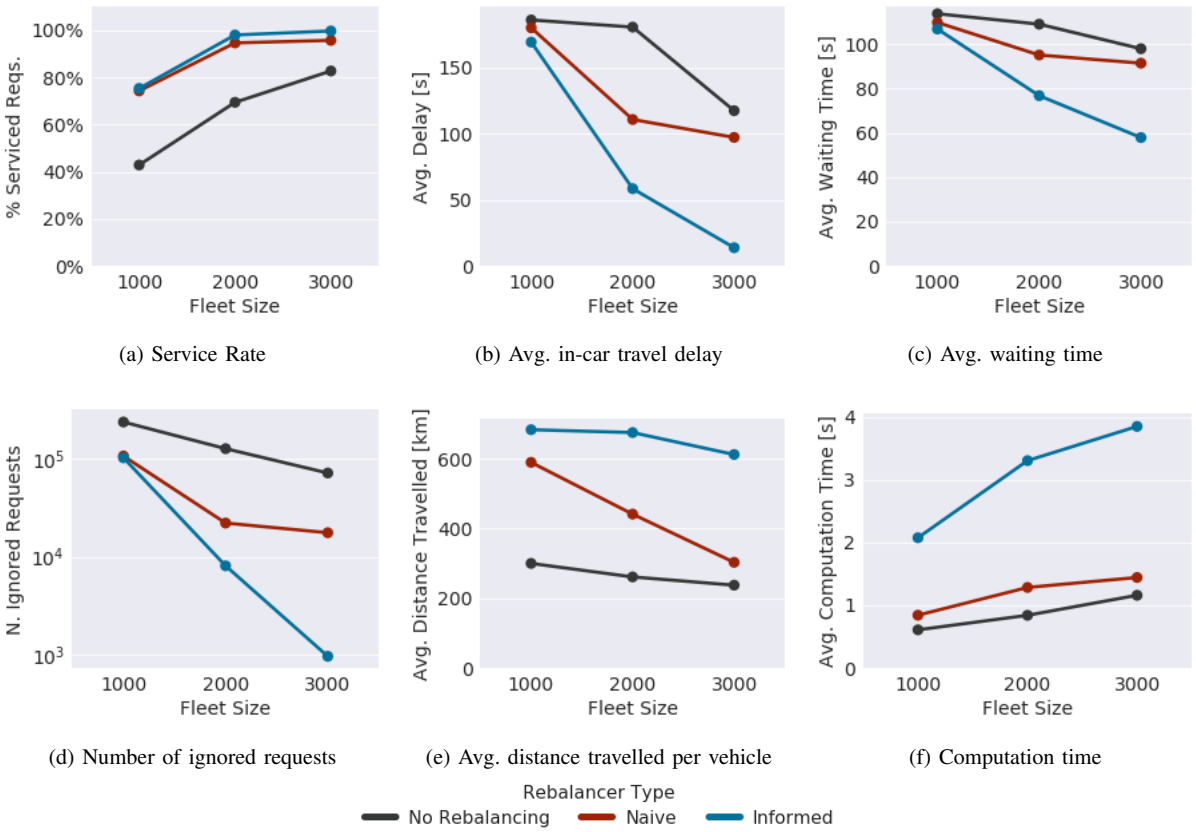
This optimization will be executed repeatedly after every time interval ψ, after vehicles have been assigned schedules to pick up and drop off requests.

V. Evaluation

We evaluate the proposed informed rebalancer using historical taxi request data from Manhattan [11] and compare the performance to the state of the art. Since the rebalancing and scheduling algorithms use timeouts to prematurely exit from the optimization, a more powerful computer can lead to much better results. To ensure a fair comparison to previous work, we reimplemented the rebalancer described in [1] and ran experiments on the same machine using the proposed informed rebalancer, the naive rebalancer from [1], without rebalancing and compared the performance of the MoD fleet.

A. Experimental Setup

For the experiments we use one day of historical taxi data from 00:00 to 23:59 on May 1^{st}, 2013. This data is publicly available and contains all taxi trips in Manhattan [11]. The



(a) Service Rate (b) Avg. in-car travel delay (c) Avg. waiting time

(d) Number of ignored requests (e) Avg. distance travelled per vehicle (f) Computation time

Fig. 4: A comparison of several performance metrics for experiments with a fleet of 1000, 2000 and 3000 vehicles and no rebalancing, naive and informed rebalancing.

data contains the origin, destination, and associated pick up and drop off time for each taxi trip. We assume the request time and pick up time to be equal since the request time is not available. The experiments are executed using a simulator that simulates the movement of the vehicles, and to which requests are added according to the historical taxi data. The vehicle routes and travel times are determined using a stored road network of Manhattan. Like [1], we estimate the travel time for each road segment using daily mean travel time computed by the method in [12] and pre-compute the shortest path for every pair of nodes in the road network. A snapshot of the visualizer for this simulator is shown in Fig 1. A computer with a 2.6 GHz (overclocked to 4.0GHz), 18 core (36 threads) Intel i9 processor and 128GB of memory was used to run the experiments.

We assess the performance of the rebalancing algorithm with a fleet size of 1000, 2000, and 3000 vehicles and a capacity of four passengers. We used a fixed maximum waiting time of $\Omega = 3$ minutes and maximum delay of $\Delta = 6$ minutes. All requests that cannot be served within these defined constraints on waiting time and delay time are ignored, and dropped from the request pool. We used 100 particles to estimate the rate of requests in each region. The vehicle locations are initialized uniformly on vertices in the road network. The assignment interval was chosen as $\psi = 30$ seconds as in [1]. This means that vehicle schedules and the assignment of rebalancing stations are optimized every 30 seconds. At assignment time, all requests are considered that have not yet been picked up. To discretize the operating area into rebalancing regions, we used a maximum reachability time of $t_{max} = 150$ seconds which produced 61 regions. The centers of these regions are shown in Fig. 3.

We evaluate two rebalancing techniques: our proposed informed rebalancing algorithm and the naive rebalancing algorithm presented in [1]. The rebalancing algorithm in [1] assigns an idle vehicle to move to the locations of unassigned requests. The assignment minimizes the sum of the distances travelled by the vehicles. We compare the results of these rebalancing techniques to a case were no rebalancing is performed.

B. Results

We collect several metrics to assess the performance of the rebalancers including the service rate, in-car travel delay, waiting time, number of ignored requests, distance travelled per vehicle, fleet utilization, and computation time. The service rate is defined as the percentage of the total number of requests that were successfully served within the waiting

Fleet Size	Rebalancer	Avg. Delay [s]	Avg. Wait Time [s]	Avg. Dist. [km]	Avg. Comp. Time [s]	N. Ignored	% Serviced Reqs.
1000	No Rebalancing	185.94	113.70	300.14	0.61	239154	42.83
1000	Naive	180.03	109.76	590.02	0.84	107462	74.31
1000	Informed	169.20	106.81	683.75	2.08	103022	75.37
2000	No Rebalancing	180.53	109.00	261.39	0.84	127796	69.45
2000	Naive	110.68	95.09	441.57	1.29	22196	94.69
2000	Informed	58.36	76.74	675.55	3.31	8108	98.06
3000	No Rebalancing	117.32	97.93	237.72	1.16	72048	82.78
3000	Naive	97.11	91.40	303.24	1.44	17669	95.78
3000	Informed	13.72	57.85	612.56	3.85	964	99.77

TABLE II: A detailed overview of the performance metrics for 1000, 2000 and 3000 vehicles for experiments with no rebalancer, and for the informed and naive rebalancer

time and delay time constraints. The in-car travel delay for a request is defined as $\delta_r - \omega_r$. The fleet utilization is defined as the average percent of the fleet with assigned schedules throughout the day. The computational time includes the time required to compute schedules, estimate demands, and compute the rebalancing assignment. These metrics are plotted in Fig. 4. The associated raw data is shown in Tab. II.

We observe that the service rate improves for all fleet sizes when using the proposed informed rebalancer rather than the naive rebalancer (See Fig. 4a). Most notably, for a fleet size of 3000 vehicles, the service rate increases by 4%. Also the proposed rebalancer achieves a higher service rate with a fleet size of 2000 vehicles (98.1%) than the naive rebalancer with a fleet size of 3000 vehicles (95.8%). This means that by switching to our rebalancing algorithm, you can reduce the size of your fleet by over 33% while maintaining the same service rate. We also see a drastic reduction in the number of requests the algorithm is not able to satisfy for all fleet sizes (See Fig. 4d). In particular, for a fleet size of 3000 vehicles, the proposed algorithm reduces the number of ignored requests by 95% compared to the naive approach.

The in-car travel delay and waiting time also benefit from informed rebalancing (See Fig. 4b and 4c). For all fleet sizes, the average in-car travel delay and waiting time decreases when using the proposed rebalancer. For 3000 vehicles, the average delay drops from 97.1 to 13.7 seconds (86% improvement) and the average waiting time drops from 91.4 to 57.9 seconds (38% improvement).

We also observe higher vehicle utilization for the informed and naive rebalancers compared to not rebalancing for all fleet sizes (See Fig. 5). The informed rebalancer achieves the highest vehicle utilization for all fleet sizes. The naive and informed rebalancers achieve similar utilization for a 1000 vehicle fleet, but for 2000 and 3000 vehicle fleets, the informed rebalancer performs much better. This can be explained by the fact that both the naive and informed rebalancer for a 1000 vehicle fleet utilize almost all vehicles continuously over the duration of the experiment.

As in [1] and [9], we observe that the advantages by using a rebalancer come at the cost of an increased distance travelled by the vehicles. This is apparent from figure Fig. 4e. This might lead to higher fuel consumption, but the initial vehicle costs and costs of potential human drivers are much lower when using a smaller fleet with comparable performance. The reason for the larger travel distances is partly

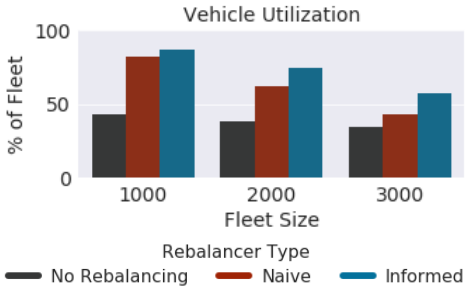


Fig. 5: Vehicle utilization for 1000, 2000, and 3000 vehicle fleet sizes. The vehicle utilization is defined as the average percent of vehicles assigned to trips over the entire experiment. For each fleet size, the vehicle utilization is measured without rebalancing, using the naive rebalancer, and using the proposed informed rebalancer

because more vehicles are being rebalanced and are moving when they are not assigned. This is also enforced however by the fact that the cost function used for assignment prefers using as many vehicles as possible with an as low as possible occupancy rate when feasible to serve requests to minimize the delay. This cost does not take into account the collective distance travelled by the vehicles.

VI. Conclusion

In this paper, we presented a method to rebalance idle vehicles in a mobility-on-demand fleet. We presented a method to optimally partition the operating area into a set of rebalancing regions, a method to compute filtered demand estimates for each region based on real-time request information, and a method to optimally assign idle vehicles to rebalancing regions using these demand estimates. Our rebalancing algorithm can be applied to human-driven or autonomous vehicle fleets. We used this rebalancer to significantly improve the efficiency of the ride-sharing algorithm presented in [1].

We demonstrated a significant improvement in fleet performance using the proposed informed rebalancing strategy over previous work. For a fleet of 3000 vehicles, we reduce the average waiting time by 37%, the travel delay by 86%, and the number of ignored requests by 95%. We show that a fleet of 2000 vehicles using the proposed rebalancer services more

passengers than a fleet of 3000 vehicles using the rebalancer from previous work.

Future work will focus on developing a method to solve for the assignment of trips to vehicles and idle vehicles to rebalancing regions in a single optimization procedure and to reduce the distance travelled by the vehicles. We also plan to expand the ride-sharing algorithm to utilize fleets with varying vehicle capacities and incorporate the existing public transportation infrastructure.

REFERENCES

[1] 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*, vol. 114, no. 3, pp. 462–467, 2017.

[2] S. Weikl and K. Bogenberger, "Relocation strategies and algorithms for free-floating car sharing systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 100–111, 2013.

[3] M. Nourinejad, S. Zhu, S. Bahrami, and M. J. Roorda, "Vehicle relocation and staff rebalancing in one-way carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 81, pp. 98–113, 2015.

[4] D. Chemla, F. Meunier, and R. Wolfler Calvo, "Bike sharing systems: Solving the static rebalancing problem," *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.

[5] S. L. Smith, M. Pavone, M. Schwager, E. Frazzoli, and D. Rus, "Rebalancing the rebalancers: Optimally routing vehicles and drivers in mobility-on-demand systems," in *American Control Conference (ACC), 2013*, pp. 2362–2367, IEEE, 2013.

[6] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.

[7] K. A. Marczuk, H. S. Soh, C. M. Azevedo, D.-H. Lee, and E. Frazzoli, "Simulation framework for rebalancing of autonomous mobility on demand systems," in *MATEC Web of Conferences*, vol. 81, p. 01005, EDP Sciences, 2016.

[8] K. Spieser, S. Samaranayake, W. Gruel, and E. Frazzoli, "Shared-vehicle mobility-on-demand systems: a fleet operators guide to rebalancing empty vehicles," in *Transportation Research Board 95th Annual Meeting*, no. 16-5987, 2016.

[9] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive Routing for Autonomous Mobility-on-Demand Systems with Ride-Sharing," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3583–3590, 2017.

[10] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, Feb 2002.

[11] B. Donovan and D. B. Work, "New York City Taxi Trip Data (2010-2013)." http://dx.doi.org/10.13012/J8PN93H8, 2014.

[12] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks.," *Proceedings of the National Academy of Sciences*, vol. 111, no. 37, pp. 13290–4, 2014.

Bibliography

[1] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics*, vol. 54, no. 8, pp. 811–819, 2007.

[2] 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*, vol. 114, no. 3, pp. 462–467, 2017.

[3] U.S. Energy Information Administration, "Energy use for transportation." https://www.eia.gov/energyexplained/?page=us_energy_transportation, 2016. Accessed: 2017-09-07.

[4] Energy Sector Management Assistance Program, *Toward Sustainable and Energy Efficient Urban Transport*. World Bank, 2014.

[5] European Commission. Directorate-General for Energy, *Keep Europe moving: sustainable mobility for our continent: mid-term review of the European Commission's 2001 Transport White Paper*. Office for Official Publications of the European Communities, 2006.

[6] World Health Organization, *Global status report on road safety 2015*. World Health Organization, 2015.

[7] NationMaster, "Motor vehicles per 1000 people: Countries compared." http://www.nationmaster.com/country-info/stats/Transport/Road/Motor-vehicles-per-1000-people, 2014. Accessed: 2017-29-10.

[8] The World Bank, "Population, total." https://data.worldbank.org/indicator/SP.POP.TOTL, 2016. Accessed: 2017-29-10.

[9] Consulting Schaller, "Empty Seats, Full Streets: Fixing Manhattan's Traffic Problem," tech. rep., New York, 2017.

[10] R. R. Clewlow and G. S. Mishra, "Disruptive Transportation: The Adoption, Utilization, and Impacts of Ride-Hailing in the United States," Tech. Rep. October, Institute of Transportation Studies, Davis, 2017.

[11] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[12] J. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo, "Vehicle Routing," in *Handbooks in Operations Research and Management Science: Transportation* (C. Barnhart and G. Laporte, eds.), vol. 14, ch. 6, pp. 367–428, Amsterdam: Elsevier, 2007.

[13] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[14] J. Cordeau, "A Branch-and-Cut Algorithm for the Dial-a-Ride Problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, 2006.

[15] J. Cordeau and G. Laporte, "The dial-a-ride problem: Models and algorithms," *Annals of Operations Research*, vol. 153, no. 1, pp. 29–46, 2007.

[16] S. Ropke, J. Cordeau, and G. Laporte, "Models and Branch-and-Cut Algorithms for Pickup and Delivery Problems with Time Windows," *Networks*, vol. 49, no. 4, pp. 258–272, 2007.

[17] K. Braekers, A. Caris, and G. K. Janssens, "Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots," *Transportation Research Part B*, vol. 67, pp. 166–186, 2014.

[18] G. Berbeglia, J. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.

[19] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proceedings - International Conference on Data Engineering*, pp. 410–421, 2013.

[20] S. Ma, Y. Zheng, and O. Wolfson, "Real-Time City-Scale Taxi Ridesharing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1782–1795, 2015.

[21] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, "Quantifying the benefits of vehicle pooling with shareability networks.," *Proceedings of the National Academy of Sciences*, vol. 111, no. 37, pp. 13290–4, 2014.

[22] F. Gheysens, B. Golden, and A. Assad, "A Comparison of Techniques for Solving the Fleet Size and Mix Vehicle Routing Problem," *OR Spektrum*, vol. 6, no. 4, pp. 207–216, 1984.

[23] R. Baldacci, M. Battarra, and D. Vigo, "Routing a Heterogeneous Fleet of Vehicles," in *The vehicle routing problem: latest advances and new challenges* (B. Golden, S. Raghaven, and E. Wasil, eds.), ch. 1, pp. 3–27, Boston: Springer, 2007.

[24] B. Golden and F. Gheysens, "The Fleet Size and Mix Vehicle Routing Problem," *Computer and Operations Research*, vol. 11, no. 1, pp. 49–66, 1984.

[25] H. Yaman, "Formulations and Valid Inequalities for the Heterogeneous Vehicle Routing Problem," *Mathematical Programming*, vol. 106, no. 2, pp. 365–390, 2006.

[26] S. Weikl and K. Bogenberger, "Relocation strategies and algorithms for free-floating car sharing systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 100–111, 2013.

[27] M. Nourinejad, S. Zhu, S. Bahrami, and M. J. Roorda, "Vehicle relocation and staff rebalancing in one-way carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 81, pp. 98–113, 2015.

[28] D. Chemla, F. Meunier, and R. Wolfler Calvo, "Bike sharing systems: Solving the static rebalancing problem," *Discrete Optimization*, vol. 10, no. 2, pp. 120–146, 2013.

[29] S. L. Smith, M. Pavone, M. Schwager, E. Frazzoli, and D. Rus, "Rebalancing the rebalancers: Optimally routing vehicles and drivers in mobility-on-demand systems," in *American Control Conference (ACC), 2013*, pp. 2362–2367, IEEE, 2013.

[30] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.

[31] K. A. Marczuk, H. S. Soh, C. M. Azevedo, D.-H. Lee, and E. Frazzoli, "Simulation framework for rebalancing of autonomous mobility on demand systems," in *MATEC Web of Conferences*, vol. 81, p. 01005, EDP Sciences, 2016.

[32] K. Spieser, S. Samaranayake, W. Gruel, and E. Frazzoli, "Shared-vehicle mobility-on-demand systems: a fleet operator's guide to rebalancing empty vehicles," in *Transportation Research Board 95th Annual Meeting*, no. 16-5987, 2016.

[33] J. Alonso-Mora, A. Wallar, and D. Rus, "Predictive Routing for Autonomous Mobility-on-Demand Systems with Ride-Sharing," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3583–3590, 2017.

[34] B. Donovan and D. Work, "New york city taxi trip data (2010-2013)," 2016.

[35] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, Feb 2002.

Glossary

List of Acronyms

SMART	Singapore-MIT Alliance for Research and Technology
TU Delft	Delft University of Technology
ILP	integer linear programming
MILP	mixed integer linear programming
VRP	vehicle routing problem
VRPPD	vehicle routing problem with pick-up and delivery
DARP	dial-a-ride problem
BC	branch and cut
RV	Request-Vehicle
RTV	Request-Trip-Vehicle
MoD	Mobility-on-Demand
TTD	time to drop-off
TTDPP	time to drop-off with predicted pick-ups
TTDPP-IR	TTDPP in combination with informed rebalancing
TTDPP-NR	TTDPP in combination with naive rebalancing
TTD-IR	TTD in combination with informed rebalancing
D-IR	delay in combination with informed rebalancing
D-NR	delay in combination with naive rebalancing

List of Symbols

Δ	Maximum delay
δ_r	Delay of a request $r \in \mathcal{R}$
κ_v	The capacity of a vehicle $v \in \mathcal{V}$
λ_g	The rate of incoming requests in a region $g \in \mathcal{G}$
$\mathcal{C}$	The set of region centres
$\mathcal{G}$	The set of rebalancing regions
$\mathcal{P}_v$	The set of requests on board of vehicle $v \in \mathcal{V}$
$\mathcal{R}$	The set of transportation requests
$\mathcal{S}$	The set of feasible schedules
$\mathcal{T}$	The set of trips
$\mathcal{V}$	The set of vehicles
Ω	Maximum waiting time
ω_r	Waiting time of a request $r \in \mathcal{R}$
π_S	The number of predicted additional pick-ups for a schedule $S \in \mathcal{S}$
ψ	The schedule optimisation interval
$\tau(q_i, q_j)$	The travel time between two locations q_i and q_j
t_r^d	The drop-off time of a request $r \in \mathcal{R}$
t_r^p	The pick-up time of a request $r \in \mathcal{R}$
t_r^r	The time of placement of a request $r \in \mathcal{R}$