ABSTRACT

Title of Dissertation: DEVELOPMENT OF AN INTEGRATED RIDE-SHARED MOBILITY-ON-DEMAND (MOD) AND PUBLIC TRANSIT SYSTEM

Liu Xu, Doctor of Philosophy, 2019

Dissertation directed by: Professor Ali Haghani
Department of Civil & Environmental Engineering

The Mobility-on-Demand (MOD) services, like the ones offered by Uber and Lyft, are transforming urban transportation by providing more sustainable and convenient service that allows people to access anytime and anywhere. In most U.S. cities with sprawling suburban areas, the utilization of public transit for commuting is often low due to lack of accessibility. Thereby the MOD system can function as a first-and-last-mile solution to attract more riders to use public transit. Seamless integration of ride-shared MOD service with public transit presents enormous potential in reducing pollution, saving energy, and alleviating congestion.

This research proposes a general mathematical framework for solving a multi-modal large-scale ride-sharing problem with real-time information. The framework consists of three core modules. The first module partitions the entire map into a set of

more scalable zones to enhance computational efficiency. The second ride-sharing module encompasses a mixed-integer-programming model to concurrently find the optimal vehicle-to-request and request-to-request matches in a hybrid network. The third rebalancing module applies advanced deep learning techniques to forecast the demand for each station and then generate an optimized vehicle allocation plan in preparation for the incoming requests accordingly.

To ensure its applicability, the proposed model accounts for transit frequency, MOD vehicle capacity, available fleet size, customer walk-away condition and travel time uncertainty. Extensive experimental results prove that the proposed system can bring significant vehicular emission reduction and deliver timely ride-sharing service for a large number of riders. The main contributions of this study are as follows:

- Design of a general framework for planning a multi-modal ride-sharing system in cities with sprawling urban layout;
- Development of an efficient real-time algorithm that can produce solutions of desired quality and scalability and redistribute the total fleet in reaction to the future demand evolution;
- Validation of the potential applicability of the proposed system and quantitatively reveal the trade-off between service quality and system efficiency.

DEVELOPMENT OF AN INTEGRATED RIDE-SHARED MOBILITY-ON-DEMAND (MOD) AND PUBLIC TRANSIT SYSTEM

By

Liu Xu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Acknowledgments

I want to express my deepest gratitude to my advisor, Dr. Ali Haghani, for his continuous inspiration, encouragement, and guidance throughout my Ph.D. study at the University of Maryland, College Park. His support and valuable advice have been the keys to the completion of this research.

My special thanks also go to the members of my doctoral examination committee, Dr. Paul Schonfeld, Dr. Cinzia Cirillo, Dr. Francis Alt, and Dr. Kaveh Farokhi Sadababi, for their constructive comments and suggestions on this work.

I would also like to acknowledge my colleagues from my research lab, Xuechi Zhang, Zhongxiang Wang, Qianlian He, Binya Zhang, Chengyang Fang, Kiana Roshan Zamir, Amir Nohekhan, and Sanaz Zahedian. The discussions with them during my lab time were delightful and memorable.

Finally, deep thanks go to my beloved husband and parents for their unconditional support and understanding during my years of doctoral study. Without their persistent encouragement, I wouldn't have been able to make it through those difficult times.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1. Background

Human society is now experiencing a series of issues coming from population growth, aging, and rapid urbanization. With the emergence of technological advances, such as smartphones, information processing, and widespread data connectivity, travel demand is evolving from an emphasis on private automobile ownership to more flexible, public and private options which incorporate shared-use and multimodal integration. Recently, new user-centric services like Uber and car2go have gained tremendous attention from the market. With their ability to allow users to access a fleet of shared-vehicles via nearly ubiquitous smart-phones, ridesharing and car-sharing services are now transforming the traditional urban mobility by providing timely transportation service to anyone with low-cost accessibility at any time. These new trends not only present a promising alternative to private car ownership but also have enormous potential with respect to air pollution reduction, energy consumption saving, congestion mitigation, and thereby improvement of the quality of human lives.

In response to the need of greater individual mobility, USDOT has proposed the concept of Mobility-on-Demand (MOD), as an innovative, user-focused approach that leverages new mobility services, integrates transit networks and operations, and connects travelers to a more effective and demand-responsive transportation system.

These include the modern transportation systems like ridesharing, car-sharing and so on. Whether and how MOD is going to reshape the current transportation system is still open for investigation.

Further, the fundamental asynchronicity between customer pick-ups and drop-offs in any ridesharing/car-sharing systems has raised the primary concern of their typically low utilization and occupancy rate. Thus, despite their potential to improve urban transportation, it is still not clear whether these systems meet expectation in their current form as an isolated entity. On the other hand, the utilization rate of public transit for commuting purposes is found to be low (McKenzie, 2010), partially due to the lack of the first-and-last-mile solution. A multi-modal transit system that seamlessly integrates a MOD service with mass transit service can benefit both sides. The hybrid system also has the potential to reduce the empty vehicle miles traveled and corresponding negative effects such as congestion and emission. Given the absence of existing efforts regarding this topic, this research seeks to fill that void by providing a rigorous assessment of the potential for integrating the MOD and mass transit systems.

1.2. Research Objective

The primary objective of this study is to develop a multi-modal integrated MOD service with a mass transit system to shift more travelers to higher occupancy traveling modes. In the long-term, this hybrid system could potentially alleviate congestion, reduce fuel consumption, and vehicular emissions. More specifically, the proposed system should be capable of achieving the following:

- Designing a proper seamless integration of MOD and mass transit system that can offer fast, reliable, and affordable multi-modal transportation;
- Incorporating ride-sharing in the first-and-last-mile MOD service without violating the requests' arrival time constraints;
- Optimizing the request to MOD vehicle assignment with the goal of achieving maximum system performance;
- Finding the optimal transfer stations and route for each customer, and if possible, match requests that could share a ride to generate higher occupancy rate for MOD vehicle;
- Managing the fleet size of the entire MOD system smartly to reduce empty vehicle mile traveled and move vehicles in preparation for the incoming demands; and
- Validating the potential of the proposed system through real-time simulation and quantitatively assessing its performance and benefits.

The proposed framework should bear the following features to accomplish all the goals mentioned above.:

- The capability to produce a real-time solution which can intelligently track the status of customer demands, vehicle occupancy, and their locations at each time stamp and update that information in a timely fashion;

- Realistic representation of the temporal and spatial characteristics of customer requests and the non-stationary network conditions in response to the time-varying demands;
- Proper formulation of the real-world operational constraints, such as hard time window constraints for commuter requests, the waiting time tolerance for the customer, and walk-away conditions for unsatisfied customers.
- Efficient fleet management strategies for MOD vehicles to serve as many requests as possible and rebalance them to reduce the total empty vehicle miles and vehicular emission; and
- Identification of proper measurements for evaluating the effectiveness of the proposed system and determining the key criteria to ensure its adequate performance.

1.3. Dissertation Organization

Based on the functions and features of the proposed system, this dissertation has been organized into eight chapters. The remaining chapters of this dissertation are as follows:

- *Literature Review:* Chapter 2 presents a comprehensive literature review of the related studies for methodologies including map partitioning, ridesharing, and rebalancing strategies to handle real-time, large-scale customer demands. Since no existing study can be directly applied to solve the proposed problem, the key factors of all relevant issues are identified and discussed in detail to facilitate this research.

- *Modeling Framework:* Chapter 3 presents the overall structure of the proposed system, which runs in an iterative fashion to handle the large-scale real-time demand with uncertainty. The key control parameters and the three core modules are presented, and their complex interactions are illustrated.
- *Module-1 Map-Partitioning:* Chapter 4 discusses the key logic used to develop the first module, map partitioning. The intuition behind its development is to solve the lack of scalability issue, which is a common challenge for the dynamic ride-sharing problem. A soft-clustering method is proposed to assign requests to their nearby station with the best accessibility in terms of both distance and MOD vehicle availability. The output of this module will be inputted into the next one, ride-sharing module.
- *Module-2 Ride-sharing Module:* Chapter 5 describes the core logic of the entire proposed study, the ride-sharing module. This chapter presents the detailed steps to solve the large-scale dynamic user demands in real-time, focusing on the formulation and the optimization approach to concurrently generate vehicle-to-request and request-to-

request matches. The critical features and key parameters in this model are also high-lightened.

- *Module-3 Rebalancing Module:* Chapter 6 presents three models to predict the future demand for each station and selects the model achieving a balance between scalability and prediction power as the model for the final application. Given that as the input, the system then applies an optimization model to generate the best vehicle relocation plan, which is intended to improve the utilization rate of the MOD vehicle and lower the unserved request ratio. The future work lies in completing this module, and key tasks are listed.
- *Numerical Results:* Chapter 7 presents two numerical cases to test the reliability and robustness of the proposed method. Numerical case one tests system efficiency under high demands for long-distance commuter trips. Numerical case two examines the benefit with the integration of the rebalancing module and confirms the system's proper performance under medium level short-distanced trips. A set of key control parameters and their impacts on the system's performance are discussed. Two systems trade-offs are also analyzed.
- *Conclusions and Future Work:* Chapter 8 summarizes the entire work with an emphasis on its contributions. Future research plans are also listed.

Chapter 2: Literature Review

2.1. Introduction

Given the absence of existing studies that can directly address the proposed topic, this chapter provides an in-depth review of relevant research efforts. The purpose is to identify the general features, strengths, and deficiencies of existing studies and shed light on this research. Most importantly, to achieve the seamless integration of MOD service which incorporates ridesharing with the mass transit service, one needs to design an efficient system to combine key methodologies to serve a unified goal. Such a system, obviously, cannot be reached without understanding the real-world intricacies, the potential interactions, and conflicts among different modules.

This review has divided all related studies based on the three core modules in the proposed work to facilitate the presentation. Section 2.2 presents an up-to-date introduction for the concept of Mobility-on-Demand and the anticipated benefits when combining its usage with public transit. Section 2.3 summarizes the pros and cons of the other demand-responsive transportation service, in specific, presents a comparative concept but with limited success after years of real application. Sections 2.4 and 2.5 review the up-to-date methodology on the topic of ride-sharing and rebalancing algorithms, corresponding to the three key modules of this study. Section 2.6 highlights the high-level objectives of the proposed study by revealing the gaps that exist in the literature.

2.2. Mobility-on-demand and Public Transit

People around the world use private cars for commuting purpose. The majority of these trips are single vehicle occupancy. Approximately 77% of all commuter trips in the U.S are single-occupant trips. (Polzin and Pisarski, 2013); similar percentages are also found in Europe (European Environment Agency, 2010). Such a low occupancy rate under high traffic demand, for example during peak hour, can cause severe congestion problems in urban areas. This issue along with the accompanying air pollution is one of the greatest challenges facing cities all around the world. The congestion in the U.S. alone costs roughly $121 billion per year, which equals 1% of the GDP (Schrank et al. 2012). This includes 5.5 billion hours lost in congestion and an extra 2.9 billion gallons of fuel consumption. There is also the potential negative externalities induced by congestion, such as the greenhouse gas emissions, travel-time uncertainty and a higher risk of accidents (Pant and Harrison, 2013; Carrion et al., 2012 and Hennessy et al., 1991).

Governments have long realized this situation and promoted public transit as its solution. However, many suburban or rural areas are not adequately served due to either their relatively low population density or the lack of economic benefits. In cities with sprawling suburban areas, the usage of public transportation for commuting purpose is very limited, e.g., less than 5% in metropolitan areas like Houston and Atlanta (McKenzie et al., 2010). Such a low utilization rate may be partly due to the absence of the first-and-last-mile solution between transit stations to riders' origins or destinations. Transportation agencies attempt to address this issue by operating a fleet of demand-responsive feeder vehicles including local taxi or small- to medium-size buses.

Recently, the wide adoption of smartphones along with the almost anywhere, anytime accessibility of cellular communication has led to the emergence of the Mobility-On-Demand (MOD) services. Leading companies like Uber, Lyft and Via in the U.S. provide users with reliable, less stressful and cost-effective MOD services that match travelers in need with drivers who are willing to provide riding services using their private vehicles upon request. The key concept behind MOD is to offer travelers mobility solutions based on the actual needs in a pay-per-trip fashion. This enables a potential shift-way from private vehicles to mobility solutions that are consumed as a service. Recently, the U.S. Department of Transportation had immense interest in the MOD service and proposed the concept of integrated transit networks and operations with typical MOD services, which could form a seamless trip chain by the integration of multiple modes of transportation and allow users to book and pay collectively for all legs of trips (Wikipedia, 2018). Unsurprisingly, this concept is brand-new and warrants investigation before its actual implementation.

While the typical MOD services present great potential to improve urban transportation and pave the way for more scalable and intelligent transportation systems, there exist a few major concerns in their current form as an isolated service. One of the primary issues is their typically low utilization or occupancy rate. One of the options to solve this is to incorporate ridesharing, as is done by services like Via and Bridj, and to a lesser extent by Uber and Lyft (Vakayil et al., 2017). Another option is to use MOD systems as a feeder to mass transit services, thus pushing more travel demand to a very high occupancy mode and using more flexible MOD service to solve the first-and-last mile accessibility problem. The integrated multi-modal transportation system incorporates both public transit and ridesharing, which in fact, complement each other.

On the one hand, ride-sharing can serve as a feeder that connects less densely populated areas to the public transit system. On the other hand, public transit can extend the reach of ridesharing and provide riders with lower cost transportation alternatives. Hopefully, attracting more riders using the service results in dealing with the low occupancy rate issue and at the same time the total empty vehicle mile traveled will be reduced as well.

Since the concept of MOD service is still under development, this field has not yet been investigated except for two pioneering works. In 2017, Vakayil et al. proposed a framework for assessing the potential of integrated autonomous vehicle MOD systems with public transit (not allowing ride-sharing) and justified its promising results with extensive simulation experiments. In that article, the authors modeled and analyzed the hybrid system while accounting for transit frequency, transfer cost and autonomous MOD vehicle fleet rebalancing strategies. The results showed that the integrated system could provide up to 50% reduction in total vehicle miles traveled. Although this study is pioneering research, its methodology and results are still preliminary in nature. For example, they avoid the pricing issue and uncertainty in

MOD vehicles' compliance rate when taking the rebalancing orders by assuming 100% autonomous vehicles.

Further, they require users to first walk to the nearest virtual hub so that they can assign MOD vehicles to pick them up at a set of preset hub locations. By doing so, the mathematical formulation of the MOD vehicle routing problem is considerably simplified, thus failing to address the biggest challenge that comes with the real-time MOD vehicle assignment and routing problem. Nevertheless, the authors also mentioned that it would be fruitful to expand their work to incorporate ride sharing, thereby operating the system at a viable price point.

Another work by Stiglic (2017) integrated ride-sharing and public transit and showed a significant enhancement for urban mobility along with an increase in public transit usage. Under the multi-modal scenario, they matched at most two riders and a driver in which the driver transports the riders to a transit station. However, public transit is only feasible for riders within the maximum acceptable walking distance of their destination, which is 0.5 mile. Also, all the riders, if using transit, are assigned to their nearest transit stations. By using this assumption, the MOD system only provides the ride-sharing option for a small number of riders, and this assistance is limited to their first-mile problem.

Moreover, the MOD and public transit systems are not optimized in unison. This is even not necessary under their assumptions since the riders are assigned to their nearest transit station to satisfy the walking distance constraint. The first-mile ride-sharing assumption also eliminates the necessity of developing efficient MOD vehicle rebalancing strategies.

In summary, there are no existing studies that address the real-time door-to-door ride-sharing problems for MOD systems to be the first-and-last-mile solution for mass transit. Nor does there exist a study that considers the optimization problem with the integration of the MOD system and public transit while allowing for realistic ride-sharing. This study proposes a rigorous methodology to fill this gap.

2.3. Demand-responsive Transportation Services

To facilitate the use of public transit system, governments all over the world have promoted a wide variety of demand-responsive transportation services for some time, where the focus of those services is often placed on people with disabilities or financially unable to have a private car. One of the common options to integrate a fixed-schedule system with an on-demand feeder service is the Demand Responsive Connector (DRC), which typically is deployed within some specific zones to transfer passengers from or to a fixed-route transit network. Several studies have been found in this area.

Koffman (2004) reviewed real-world examples of DRCs implementation in several cities in the U.S. and concluded that it had been one of the most popular types for assisting flexible transit services.

Regarding the decision-making aspect, some studies investigated conditions where the transit operator could choose a demand-responsive service rather than provide a fixed-route feeder system, in terms of both service quality and operating cost. An example is Quadrifoglio's work in 2009, where a simulation model was developed and the threshold to switch between a demand-responsive and a fixed-route feeder service was found to range from 10 to 50 passengers per square mile per hour. Similar works are Li et al. (2010) and Qiu et al. (2015).

Li and Quadrifoglio (2011) proposed a continuous approximation model to define the optimal zone for implementing DRC service. Along these lines, Lee and Savelsbergh (2014) tested different feeder services in a zone with multiple transfer stations and where passengers can be dropped off at any of those to meet their desired service. Further, the results demonstrated that a demand-responsive system could offer substantial cost saving compared to fixed-schedule service, especially when transit services are frequent and stations are close to each other.

In summary, effective integration of the ride-sharing MOD system and a scheduled public transit system have the immense potential to extend public transit system coverage, which has many societal and environmental benefits. However, the biggest challenge is that the transfer from one mode to the other must be seamless and efficient and without long wait times for riders. Consistent transfer can only be available with efficient optimization technology.

2.4. Ridesharing

Ride-share providers globally are offering convenient real-time service to compete with the traditional private vehicle transport mode. To broaden its appeal, ride-sharing must be easy to access, have lower cost, be able to provide door-to-door service, flexible, efficient and reliable. However, MOD and ride-sharing are two different concepts, though. MOD indicates an on-demand transportation mode compared with private car ownership or planned transportation service. It can be cooperating with ride-sharing but is not limited to it. The development of algorithmic approaches for optimally matching drivers and riders in the real-time fashion and assigning drivers with the best route to pick up multiple riders play key roles in ensuring the ultimate success of ride-sharing. This section systematically outlines the challenges that arise when developing technologies to support ride-sharing and summarizes the advantages and limitations of related models in the academic literature.

The ride-sharing problem is a special case of the *dial-a-ride* problem (DARP). Typical DARPs consist of finding the optimal routes and schedules to transport a set of passengers with a specific origin-destination and time-window constraints regarding their pick-up and drop-off times. Besides, additional inconvenience constraints might apply when transporting passengers depending on the specific problem statement. The objective function is usually set to minimize the total system cost. Although formulations of DARPs can take a variety of different forms due to the variance in service quality elements, the common attributes are waiting time, maximum ride time, and the time difference between desired and actual arrival time. The ride-sharing problem, despite the considerable similarities, differs from the conventional DARPs in some respects, as pointed out by Agatz et al. (2012). They distinguishingly modeled the case where the vehicles are independent of the normal service provider. Such an assumption led to additional dynamics that the origin, destination and the appearance of the drivers are uncertain. Therefore, the methodologies for DARPs cannot be universally applicable to the ride-sharing problem.

Due to its complexity, the ride-sharing methodologies can be classified in many ways given different criteria. Based on the number of drivers and riders participating in the ride-sharing activity, Agatz et al. (2012) summarized existing efforts into four categories, encompassing *single driver-single rider, single driver-multiple riders, multiple drivers-single rider, and multiple-drivers-multiple riders* assignments. In our case, we are considering the multiple drivers-multiple riders ride-sharing methodology.

Other researchers, like Yan and Chen (2011), made different classifications based on the deviation of riders' routes. They distinguished four main types of ride-sharing problems, including *1) one origin to one destination (OOOD); 2) many origins to one destination (MOOD); 3) one origin to many destinations (OOMD); and 4) many origins to many destinations (MOMD).* With the technological advancement of mobile phones, the MOMD model has undergone great popularity recently. A mathematical example is the work of Di Febbraro and Gattorna (2013). They used a discrete model to find the best match and route between drivers and riders. The optimization objective is to achieve the minimized deviations between passengers' desired and actual arrival times. Other examples include real-world applications, like Uber Pool, Lyft line, Sidecar, Flinc and Carma that provide dynamic ride-sharing options for interested users if applicable. Users can access and request ride-sharing via their smartphones and pay a lower cost than riding alone.

As for the OOMD models, Chen and Liu (2010) proposed the vehicular ad-hoc network optimization model targeting on maximizing the fuel-savings. To enhance their models' computational efficiency, the entire map was divided into several grids, and they matched the passengers with the same destination grids as well as all its neighboring grids. This is one of the examples of considering network topology by partitioning the roadmap.

Various studying are found in solving the MOOD models. For example, Baldacci et al. (2004) developed both exact and heuristic approach to address the MOOD problem. Tao (2007) implemented a taxi ride-sharing system that selects the shortest path through enumerating over all possible combinations.

Godofalvi and Pedersen (2009) took a different perspective and approached this problem by maximizing the financial benefits. Their algorithm finds the best combinations of ride-sharing partners in a group of maximum taxi sharing size, where "best" is defined as sharing the longest route. The algorithm then selects the one with the largest overall saving among all combinations. The limitation of their work is obvious. Under their optimality condition, riders may experience larger waiting times than those minimizing total detours. Also, they assume the taxi-fare has a linear relationship with the distance, which possibly leads to sub-optimal solutions in a real case. This is because the cost is more relevant to the travel time, which is highly depend on traffic condition ranther than the linear distance.

In the same year, Geisberger and Luxen (2009) presented an algorithmic solution to match ride-sharing offers and requests for arbitrary starting and destination points. They first add each new request to the existing request pool and find the combinations with the shortest detour. Through experiments, their detour method runs faster than the 2k+1 distance calculations. In terms of static demands, the algorithm can handle large-scale requests with potentially hundreds of thousands of users each day. However, their goal is to find the optimal solution via an exhaustive search, which limits the algorithms ability to solve the dynamic ride-sharing problem.

Continuing this track, Geisberger (2012) developed a spatial, temporal and hierarchical decomposition method to solve the challenges that come with the mathematical formulation and the solution algorithm for dynamic ridesharing matching problem. He proposed the Three-Spherical Heuristic decomposition model (TSHDM) and assessed the quality of the solution compared to the exact solution. A case study was then performed on the network of the northwest metropolitan area of Baltimore city. The simulation results confirmed the effectiveness of the proposed model. The study also shows that the shared ride rate is highly dependent on origin and destination locations of the participating riders.

Horn (2002) published an article which describes a software system designed to manage the deployment of a fleet of demand-responsive passenger vehicles such as taxis or variably routed buses. In this article, a model with an objective of minimizing additional travel time or maximizing a surrogate for future fleet capacity was presented.

Agatz et al. (2011) developed a heuristic-based approach that minimizes the total system-wide vehicle miles incurred by system users and their individual travel costs. Based on the 2008 demand data in metropolitan Atlanta, their simulation revealed that sustainable populations of dynamic ride-sharing participants might be possible even in relatively sprawling urban areas with many employment centers.

Ma et al. (2013) proposed a two-stage algorithm to solve the large-scale taxi ridesharing problem. They first proposed a taxi searching algorithm using a spatiotemporal index to quickly retrieve candidate taxis that are likely to satisfy a user query. A scheduling algorithm was then used, which checks each candidate taxi and inserts the query's trip into the schedule of the taxi that satisfies the query with minimum additional incurred travel distance. Their simulation results showed a 25% increase in the number of taxi users with a 13% travel distance saving compared with a no-ride-sharing case.

Another approach is the one described by d'Orey (2009), which proposed a two-stage taxi-sharing method, including a customer algorithm and a taxi algorithm. The customer algorithm collects the riders' spatiotemporal information and gives it to the taxi within a given range. Their optimization goal was to minimize distance traveled.

Other than the classical centralized optimization approaches, researchers also employ agent-based methods to model dynamic ride-sharing problems. Mes et al. (2007) compared an agent-based vehicle-scheduling model with classic look-ahead heuristics. They simulated 20 autonomous vehicles and demonstrated that the multi-agent model has a more stable performance in response to the fluctuations of demand and supply. Regarding agent-based models, a question that arises in the limitation of the individual information on the system performance. Winter and Nittle (2006) addressed this issue by developing a system where agents are matched via short-range communications. They found that a local communication system doesn't perform significantly worse than the one that can access global information.

Map-Partitioning Approaches

One common technical bottleneck for the dynamic ride-sharing problem is the lack of scalability. Two approaches are typically used to resolve this issue. One is through parallel computation to balance the workload and the other is to partition the road network into smaller regions.

There were early attempts to reduce the size of the pickup and delivery problem (PDP). Jaw et al. (1986) decomposed the entire problem by dividing the time horizon into intervals and then grouping the clients based on their desired time window.

Nalepa et al. (2015) proposed to partition the search space in their parallel guided ejection search algorithm to minimize the fleet size for pickup and delivery problems with time windows. They confirmed that this technique could decrease the convergence time without affecting the quality of results.

For the category of enhancing computation efficiency through map partitioning, Mohring's work (2005) can serve as an example. He compared several partition schemes in terms of speeding up the Dijkstra algorithm. These attempts included dividing the large network into grids, Quad-tree, Kd-tree, and METIS with a goal of generating a balanced number of vertices or edges in each sub-map. Quad-tree is a data-structure, which stores points in a plane and typically generates fast access to the nearest neighboring points. However, it fails to take the distribution of the points into accounts, which often is the case for a real-world Dijkstra problem. To address this issue, the author further applies Kd-tree to develop a partitioning algorithm that can be extended to more general subdivision schemes. The METIS partitioning, introduced by Karypis and Kumar (1998), is a fast method to partition a graph into R almost equally sized sets with a noticeably smaller number of arcs than in the other partitioning methods. Their results show that the algorithm speeds up the computation time compared with the standard Dijkstra algorithm, and the speedup increases with the size of the graph. Among all examined partitioning schemes, Kd-tree and METIS accelerate the computation the most.

Mitrović-Minić et al. (2004) applied a partition-based approach to solve the PDP. They formed spatial-temporal 3D boxes with the rectangular base representing the geographical area and height defining the time span at which zone is serviced by the vehicles. Requests for pick-up and drop-offs are added to the existing zone, and once the zone expands over a certain size, it will split into two.

Sáez et al. (2008) employed the Fuzzy C-Means method to divide the entire map into multiple, homogeneous and non-overlapping sub-zones. The partition is made based on the historical origin and destination patterns.

Wei et al. (2010) partitioned the network with an objective of balancing the computation workload so that the simulation runs faster. Besides dividing the road network into homogenous zones, Gonzalez et al. (2007) employed a hierarchical path algorithm. They classified the roads into different categories such as highways, main roads, etc. Those roads divide the entire network into varying hierarchical levels.

More recent work is the one by Pelzer and Xiao (2015), which presented a partition-based matchmaking algorithm for dynamic ride-sharing. Their method looks at solving the large-scale and highly dynamic ride-sharing system that classical approaches often fail to address. The road network is divided into distinct sub-zones based on the topology that optimizes their shared-utilization.

2.5. Rebalancing

Due to the fundamental imbalance of the customer pick-up and drop-off in any car-sharing or ridesharing system, a fleet management strategy is essential to ensure adequate performance. For example, Pavone et al. (2012) showed that rebalancing in most of the autonomous MOD systems is necessary to prevent unbounded customer queues. The two main categories of vehicle relocation strategies found in the literature: *operator-based* and *user-based.*

Some existing studies that aim at maximizing the profitability of the car-sharing/ ride-sharing service can be classified into the operator-based category. Corriea and Antunes (2012) applied mixed-integer programming models to locate one-way car-sharing stations where vehicle balancing issues exist. With a goal of maximizing net revenue, they compared three different trip selection schemes under the same depot features. However, the model did not allow for integrating the relocation operations due to the scalability problem. Also, by maximizing the system profit, Diana et al. (2014) proposed a real-time vehicle relocation scheme for a one-way car-sharing system. They compared two different rebalancing strategies. Both, when implemented, showed a significant increase in profit.

Examples of the user-based category include the work by Smith et al. (2013), who optimally routed the rebalanced vehicles to minimize the total number of vehicles performing this task. Under their assumed Euclidean network topology, they found that only between 1/3 to 1/4 drivers are needed from the original fleet size. Similar research can also be found in Fan et al. (2008) and Kek et al. (2009).

More recently, Spieser et al. (2015) presented a rebalancing scheme for autonomous MOD system. They formulated a MIP to minimize the total travel time for vehicles conducting the rebalancing task until the desired distribution. Through asimulation using car2go rental data, their results revealed that rebalancing could significantly reduce the needed fleet size.

Besides the optimization approach, some researchers explore this problem in other ways. For instance, Fagnant and Kockelman (2014) employed an agent-based simulation for a shared vehicle system to test different rebalancing strategies. They proposed that one shared AV can replace roughly eleven conventional vehicles with a total vehicle mile traveled increase of 10%. In 2015, they further extended this work by investigating the potential of shared AV operations. With a 1.3% penetration rate, comparable vehicle reduction ratios and increases in total vehicle miles were reported. Additional simulation-based examples can be found in (Papanikolaou, 2011; Barrios, 2012; Barth and Todd, 1999; Kek et al., 2006 and Efthymiou et al., 2012)

Zhang et al. (2014) provided a theoretical queueing approach for modeling a network of self-driving vehicles and equalized the fleet availability across all stations. Through a simulation testing using New York City taxi data, they demonstrated a 40% reduction in required fleet size when applying rebalancing strategies.

2.6. Summary

Based on the extensive literature review reported in this chapter, it is clear that:

- Although several studies have researched the ride-sharing problem, they mostly emphasize developing methodologies involving a single transportation mode, which typically is the regular passenger car. Little effort has focused on studying multi-modal ride-sharing.
- Regarding the optimization for dynamic ride-sharing, much literature addressed this focusing on testing their proposed models on small-to-medium sized random requests. Few publications have attempted to deal with this issue under the context of large-scale random demands which is the key challenge for this topic, and where the classic approaches usually fail.
- Due to the asymmetric nature of the customers' pick-up and drop-off locations, an effective rebalancing strategy needs to be developed and combined with the ride-sharing control module. However, most of the existing studies dealing with taxi-based ride-sharing ignore this necessity and assume perfect vehicle allocations instead.

- Very few models have attempted to solve large-scale multi-modal ride-sharing with the fleet management function. Although this idea has been promoted by one paper, the authors failed to explore this issue thoroughly as they only serve requests with the destination right at the metro stations.
- As an emerging technology, MOD has immense potentials to improve urban mobility and reduce pollution in the long run, especially when combined with the under-used public transportation system. This concept still requires comprehensive investigation and huge attention.

Given the urgent need for supporting the enhancement of urban mobility and the evident limitation of the existing literature, this study aims at bridging the gap by addressing the above-mentioned critical concerns individually. The framework of the proposed system will be presented in the next chapter, and the modeling details within each core module will be introduced in the following chapters.

Chapter 3: Modeling Framework

This chapter introduces the proposed multi-modal ride-sharing system designed for dealing with the large-scale dynamic requests for rides in an urban area. The proposed system is generic in nature, including all the essential components and their interrelations under any applicable urban layout with a massive public transit system. To simplify the presentation, this research uses metro transit as an example of mass transit. The next section first illustrates the entire models' key functional modules, where the map-partitioning module assigns MOD vehicle and riders to their best station. The ride-sharing module as the second one, also the core for the proposed system, takes the output from the previous module and is responsible for both matching available MOD vehicles and in-queued requests and matching the riders who can be ride-shared without violating their arrival hard-time windows. The third module, rebalancing, is to allocate the fleet best to serve the incoming demands. Note that the entire system is running in a real-time fashion, which means it will update and re-run all the modules in each iteration. Section 3.2 discusses all the critical issues associated with the development of each module and their complex interactions. Section 3.3 highlights the specific objective of each module. Section 3.4 summarizes all the key components in all modules.

3.1. Framework and Key Components of the Proposed System

Figure 3-1 presents the structure of the entire proposed system, highlighting all the essential modules and the key interactions among all the components in this research. Note that the process presented in the figure will be executed recursively to fit the real-time context. A brief description of each module and their complex interrelations is given below.

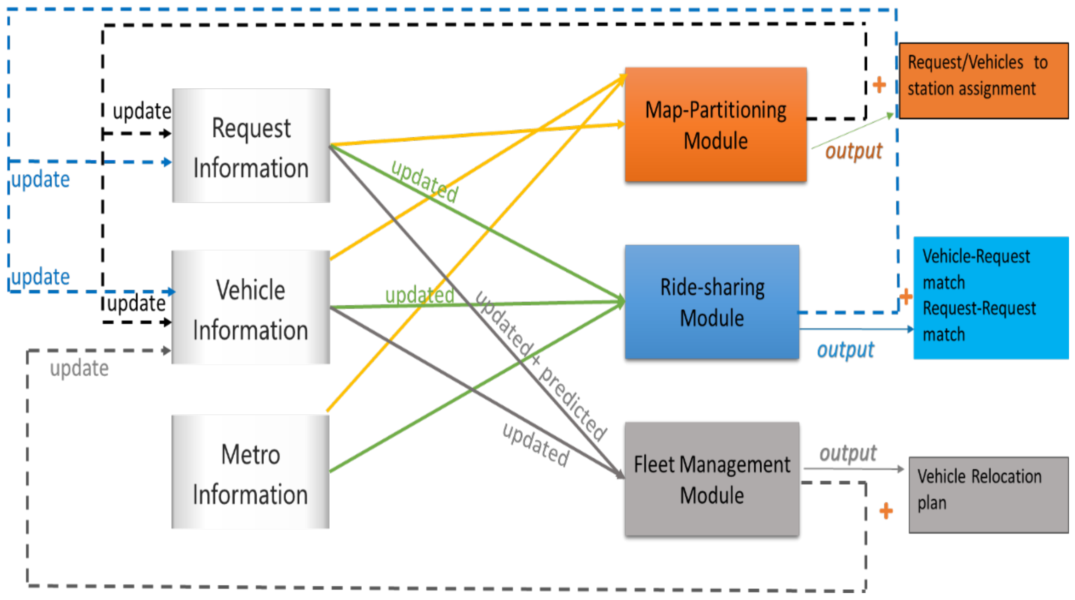


Figure 3- 1.The framework of the proposed system

- Input Datasets: As shown on the left-hand side of Figure 3-1, the overall input dataset can be categorized into three types. They are request information pool, vehicle information pool, and metro information pool. All the information stored in those pools will be updated with the output generated by three key modules successively and iteratively.
    - o *Request information*: collects the information of requests that occurred in the current interval time, which includes the origin-

destination longitude and latitude, customer departure and arrival times and the number of passengers.

    - *Vehicle information:* keeps track of each vehicle's status, including location, remaining capacity, and the trip or rebalancing task it is executing at each time stamp.
    - *Metro information:* records the schedule of each metro line and the location of the metro stations (longitude and latitude). More specifically, the schedule stands for the arrival and departure times of each metro lines operating at each station.

- Key functional modules: There are three key functional modules in the proposed model. They are Map partitioning module, Dynamic ride-sharing module, and the fleet management/rebalancing module. These three modules run successively, which means the latter module takes the former ones' output information as part of its input. The key input and output of each module and their interactions are introduced in the following.
    - *Map-Partitioning Module:* This module serves as the key to partition the large map into smaller, comparable sizes to enhance the computation efficiency of the large-scale network problem presented in the next module. At the beginning of each iteration, the request, vehicle and station location information will be inputted into this module, which using its methodology, assigns the requests,

and vehicles to each station-based sub-map. The latter module will generate solutions within each sub-map.

- *Dynamic Ride-sharing Module:* This module is the core of the entire system, which is responsible for producing the match between the vehicles and requests, the request and request, and the request and metro schedule. It takes the output from the map partitioning module together with all the existing information as its input, generates the detailed trip list for each available vehicle, and assigns the riders to the metro schedules without exceeding the requested arrival time. Note that a rider request may not be matched with any vehicle at the current iteration time, and if so, it will remain in the unserved request pool and will re-enter the matching process in the next iteration until it either is served or is declined. It is also possible a rider may not find any other riders with whom to share a ride, and if that is the case, the algorithm will let him or her ride alone. The generated results will be used to update the vehicle and request information pools. Specifically, the available vehicle list will be updated as well as their corresponding remaining capacity. The requests assigned to certain vehicles and the ones declined by the system (violating time constraints) will be removed from the unserved request pool.

   - *Rebalancing Module:* This module generates the demand forecasting for each station and returns a set of vehicle relocation plan and then is applied to update the vehicle information pool. It takes the input from the updated request and vehicle information pools. Also, there only exists two status for a vehicle, either assigned to a certain station or en route from one station to another (in other words, performing a rebalancing task). The vehicles which are undergoing rebalancing tasks will not be assigned to serve any rider requests.

From the description, one can see the interactions among different modules are frequent and complex. Although the three modules run in an orderly manner, they are in fact dependent on each other due to the system's recursive nature. The request and vehicle information pool serve as the media for their interactions.

3.2. Critical Issues

To facilitate the illustration, this section uses the scenario when MOD vehicles are integrated with the public metro system to identify all the critical issues associated with the proposed methodology. During the service time, the following information will be collected in real-time:

- Requests information: origin and destination, departure and arrival time and the number of passengers for each request.

- Vehicle information: number of vehicles in the operational fleet size, their locations, and their capacity.
- Metro information: Metro stations' location and the schedule of each line that runs at each station.

The system collects the received customer requests and adds them into the request pool. Together with the information for vehicles and metros, they will be inputted to the map partitioning module to initiate the whole procedure. Typically, for a metro network, more stations are deployed in more densely populated areas such as central business zones at the city. On the other hand, when approaching the less densely areas such as suburban areas, fewer stations are constructed. To ensure the requests are assigned to the best stations, the following issues need to be considered:

1. How to assign requests to the station with sufficient accessibility and satisfy its requested time window? This study applies a station-based partitioning based on the distance measure to address this.
2. How to select the best station for each rider when several stations are nearby, and their relative distance to the customer's location is not significantly different? This study proposes a soft-clustering method, which enables such requests to belong to multiple stations with comparable accessibility o solve this issue.
3. How to capture the randomness of the requests since they will occur at any time and any place? This study employs the recursive procedure that allows the map-partitioning to update itself at every iteration time.

Then after running the map-partitioning module, the entire map will be partitioned into a set of smaller sized ones. The ride-sharing module works on each of the sub-map. The following critical issues are anticipated:

1. How to match the vehicles with the remaining capacity to the unserved request? How to determine whether multiple riders can be ride-shared?
2. If several riders can share a ride, which vehicle will be assigned to serve them and in what order should the vehicle pick them up and drop them off?
3. How to concurrently optimize the potential metro schedules, MOD vehicle assignment and the real-time ride-sharing for both first-and-last-mile trips given a customer request?
4. How to ensure that when a rider arrives at the destination station, there is another MOD vehicle assigned to serve him or her to complete the last-mile trip?
5. How to optimize the problem to give adequately good results in terms of both the service provider and the customers?

The detailed solution to these issues mentioned above will be discussed in the later chapters.

Considering the fundamental imbalanced nature of customers' pickup and drop-off locations, almost every MOD system requires necessary rebalancing to prevent unbounded customer queue up (Spieser et al. 2016). This study also recognizes such need so that the following key tasks need to be fulfilled by the fleet management module:

1. How to relocate the available vehicles most efficiently?
2. How to predict the incoming demand accurately for each station? Also, to be applied for real-application, how can the model achieve a balance between computational efficiency and the prediction power?
3. How to optimize the vehicle relocation plans based on the demand forecasting results? What will be the objective from a system designer point of view?

The detailed methodology and key considerations for rebalancing module will be discussed in Chapter 6.

3.3. Key Objective for Each Module

To provide a theoretical basis for this seamlessly integrated MOD system, the key objective of each module is summarized below:

1. *Map-Partitioning Module:* This module assigns the requests to each station and partitions the entire map into station-based zones to enhance the scalability of the next ride-sharing module.

2. *Ride-sharing Module:* Grounded on the results of the previous module, this module generates the vehicle-to-request, request-to-schedule, and request-to-request matches to form the ride-sharing solution. The optimization model is to maximize the number of served requests.

3. *Rebalancing Module:* Based on the previous two modules, this module predicts the station based future requests and generates the optimized vehicles relocation plan grounded on that. The objective function of the rebalancing optimization model is to minimize the total travel time for MOD vehicles conducting rebalancing task.

3.4. Summary

This section provided an overview of the proposed system with the highlights of the three core modules and their corresponding key components. To realistically address the uncertainty of a MOD system, this study develops a three-module interactive approach to capture the demand and supply side updates, and this also enhances the model's scalability. Due to the complexity of the targeted problem, one cannot solve it without considering the compound interactions among all three modules. A detailed discussion of these critical issues will be provided in the following chapters accordingly.

Chapter 4: Module-1 Real-time Map Partitioning

This chapter presents the methodology for efficiently partitioning the large map into a set of smaller and scalable zones. The intuition behind this is to solve the key challenge typically presented for any dynamic ride-sharing problem, which is the lack of scalability. To achieve this goal and to address the critical issues discussed in the previous chapter, the proposed map-partitioning logic should bear the following features:

1. Zoning the large-scale demands and MOD vehicles efficiently to the stations with the best accessibility;
2. Being flexible enough to accommodate the randomness of the real-world demand; and
3. Allowing some degree of the zone overlaps to ensure each sub-map is of sufficient size, even at the area where several stations are crowded in one small geo-region (e.g., central business zone in one city).

To acquire these features, the modified fuzzy-c means zoning method is proposed. The FCM clustering is known for its ability to assign data points to one or more clusters. With the recursion of the map-partitioning algorithm, this method can generate an updated partitioning plan to accommodate the randomness of request patterns, in terms of their occurrence frequency and the uncertainty of their locations. Further, since the goal of this module is to assign requests to the metro stations with the best accessibility, the modified Fuzzy C-Means is clustering the requests based on the station locations. Section 4.1 introduces the reasoning for choosing a soft-clustering method. Section 4.2 presents the proposed methodology in detail, and section 4.3 summarizes this module in a high-level.

4.1. Soft Clustering

In the context of this study, the map partitioning is also clustering the demands into several smaller groups so that the scalability problem can be avoided. Clustering analysis involves assigning data points such that the data in the same cluster are as similar as possible, while the ones belonging to different clusters are as dissimilar as possible. Soft clustering, which is also referred to as fuzzy clustering, is a form of clustering in which each data point can belong to one or more clusters. In the case of the map partitioning, the similarity measures are, mostly, based on distance.

Fuzzy C-Means (FCM) is one commonly used soft-clustering technique in which a dataset is grouped into *n* clusters with every data point belonging to every cluster to a certain degree. For instance, the data points which lie closer to the center of a cluster will have a high degree of membership to that cluster, and another data point far away from the center will be assigned a lower membership to that cluster. This algorithm was first developed by J. C. Dunn (Dunn, 1973) and improved by J. C. Bezdek (Bezdek, 1981) later on. The algorithm for FCM is very similar to K-means clustering, except for the objective function. There is an additional membership value W_{ij} and the fuzzifier *m,* which determines the level of cluster fuzziness. A large *m* will result in smaller membership values.

The following section will introduce the detailed methodology in the proposed modified Fuzzy C-Means zoning.

4.2. Modified Fuzzy C-Means Zoning

Methodology for Zoning Requests

Instead of calculating the centroids for all clusters at each iteration, the proposed algorithm fixes the station location to be its centroids and aims at minimizing the following objective function:

$$Min \sum_{i=1}^{n} \sum_{j=1}^{s} w_{ij} \| x_i - st_j \|^2 \qquad (4\text{-}1)$$

Where $X = \{x_1, x_2, ...x_n\}$ is a finite set of n requests, ST = $\{st_1, st_2, ...st_n\}$ is a set of stations, and $w_{ij} \in [0,1]$ denotes for the degree of membership for x_i in the cluster *j*. This function takes the absolute distance between the measured requests and the stations and tries to minimize the sum of differences through an iterative optimization. By the end of each iteration, the membership matrix W will be updated by:

$$w_{ij} = \frac{1}{\sum_{k=1}^{s} \left(\frac{\| x_i - st_j \|}{\| x_i - st_k \|} \right)^{\frac{2}{m-1}}} \qquad (4\text{-}2)$$

Where *m* is the fuzzifier, which is any real number greater than 1. It is the hyper-parameter that controls how fuzzy the cluster will be. The higher the *m,* the fuzzier the cluster becomes. It is not desirable to set *m* to be either too large or too close to 1 because the former will lead to approximately equal membership in all cluster and the latter will lead to absolute membership as the one produced by a hard-clustering method (membership equals near 1 or 0). After reviewing several relevant studies, there isn't a systematic way of determining the optimal value for *m.* The choice of the degree of fuzziness can be the following: 1) the number of clusters to which each point belongs can be set based on expert knowledge and 2) the degree of fuzziness is assigned either a widely used value or one can select a value between the upper and the lower level of fuzziness if theoretical ranges are available from similar previous efforts (Ozkan and Turksen, 2007).

Most previous efforts suggest that a value of *m* around 2 generally performs well through empirical experiments (Pal and Bezdek, 1995; Fadili et al., 2001; Ozkan and Turksen, 2007; Zhou and Fu, 2014). In practice, 2 is the most commonly adopted choice for *m* (Bezdek, 1981; Hruschka, 1986; Hathaway and Bezdek, 1993).

Hence, this study adopts 2 as the degree of fuzziness for the proposed algorithm. The termination condition for this algorithm is:

$$Max\ x_{ij}\{|\,w_{ij}^{iteration\ \mathrm{n+1}} - w_{ij}^{iteration\ \mathrm{n}}\,|\} < \xi \tag{4-3}$$

Where ξ is the predefined termination threshold that typically ranges between 0 and 1, and *n* is the iteration. This procedure converges to a local minimum of the objective function. The pseudo-code of this algorithm is presented in Figure 4-1. After the termination of the algorithm, the membership matrix W is generated for all the requests, in which row *i* indexes the membership for request *i* to belong to all the stations. However, such fuzziness is not able to reduce the computation workload for the ride-sharing module, which relies on the outcome of this stage.

Most of the time, the real world demand pattern is unevenly distributed, which tends to have a high density in the central business zones and low density in other areas. For a request located very close to a certain station, the membership index for it to belong to this station must be dominating the other station memberships. In this case, the fuzziness does marginal benefits to this request. However, the fuzziness makes more sense for the requests located in between different stations, which in other words, the stations surrounding them are very comparable in terms of the distance measure. As presented in Figure 4-1, after running the standard FCM algorithm, the proposed method further picks up the requests with similar enough largest membership indexes, whereas those requests with single dominating membership index are directly assigned to the corresponding station. From the experimental results of the simulation, this study adopts 0.1 as the threshold for ∂, as it produces the desired fuzziness of the requests assignments.

That is saying, the requests with several stations of similar distance are directed into the next step to compare the availability of the MOD fleet. For requests with the direct assignment, this information will be inputted to update request information pool and then fed into module 2- the ridesharing algorithm.

However, for those requests with multiple stations of similar distance, the algorithm further takes into consideration the number of available MOD vehicles at each station using the following:

$$u_{ij} = w_{ij} + a * V_j \qquad (4\text{-}4)$$

```
The Modified Fuzzy C-Means Zoning
Begin
    Initialize the matrix $W^0$, $\xi$, maxIterations, m. And choose Euclidean distance as
    the measure of distance;
    Fix centroids by inputting the coordinates of the selected stations
    for t=1 to maxIterations do
      update membership matrix W using Eq. (4-2)
      calculate new objective function using Eq. (4-1)
      if Eq.(4-3) is True then
        break;
      else
        continue
    end for

    for each rider request i do
      rank its membership indexes in a decreasing order
      if $w_{ij}^{largest} - w_{ik}^{2nd-largest} > \partial$ then
         assign request i to station j with $w_{ij}^{largest}$
         end for
      else
         adjust the membership matrix W using Eq. (4-4)
         assign request i to station j with $u_{ij}^{largest}$
         end for
End
```

Figure 4- 1. Pseudo-code for the proposed Modified Fuzzy C-Means zoning

The U matrix stores the adjusted membership index by adding a weighted vehicle number index V_j to the W matrix. And a is the weight for vehicle numbers, which ranges from 0 to 1. As shown in Figure 4-1, the program then takes the stations with the largest u_{ij} as the assignment for each request. This information will then be given to the next module for ride-sharing.

Zoning MOD vehicles

Different from the Zoning for Request which needs to be executed at each iteration, Zoning MOD vehicles only applies to initiate the entire program at the very beginning. This section employs the same algorithm presented above except for those adjustments. The pseudo-code for the MOD vehicle zoning setup is shown below in Figure 4-2.

```
Initial Zoning for MOD vehicles

Begin
    Initialize the matrix $W^0$, $\xi$, maxIterations, m. And choose Euclidean distance as
    the measure of distance;
    Fix centroids by inputting the coordinates of the selected stations
    for t=1 to maxIterations do
      update membership matrix W using Eq. (4-2)
      calculate new objective function using Eq. (4-1)
      if Eq.(4-3) is True then
        break;
      else
        continue
    end for
End
```

Figure 4- 2. Pseudo-code of the MOD vehicle zoning setup

The MOD vehicles initial location will be inputted to the algorithm; the zoning will also be calculated based on the metro station location. Once the termination condition is met, the MOD vehicles will be assigned to each station. This assignment will remain unchanged unless the rebalancing module sends them rebalancing orders. Note that the request-zoning step also depends on these results, so this section needs to be performed before the first iteration. Moreover, it only needs to be executed once for the entire service duration.

4.3. Summary

This chapter applies a modified station-based Fuzzy C-Means clustering method to reduce the size of the searched map to enhance the computational efficiency encountered in solving real-time dynamic ride-sharing matches. At each iteration, this module will partition the entire map into a set of the more scalable regions and explore the ride-sharing opportunities within each sub-zone.

Chapter 5: Module-2 Ride-Sharing Module

This chapter presents the methodology for the core part, ride-sharing module. The proposed system can only be effective in congestion alleviation when a large number of commuters are shifted from their private car usage to the proposed multi-modal transit mode. For this purpose, this module is developed to solve the large-scale real-time ride-sharing problem together with the metro schedule selection. The rigorous methodology can then serve as a fair basis to quantitatively assess the potential benefits of the proposed system, which is the key contribution of this study. As summarized in the literature review section, the impact of large-scale multi-modal ride-sharing hasn't been investigated in the existing literature.

The proposed method must acquire the following features to serve the purpose of scientifically quantifying the potential benefits:

1. Be able to realistically model the real-world multi-modal ride-sharing problem, especially when dealing with high demands;
2. Be able to accurately translate the spatial-temporal ride-sharing problem into a graph-theoretic representation and be able to generate efficient solutions in a timely manner;
3. Be able to concurrently optimize the ride-sharing problem together with the selection of the best metro schedule;
4. Be able to rigorously develop a general theoretical framework which enables investigation of the fundamental interactions among a set of key

control parameters and understand their impacts on the system performance; and,

5. Be highly general; the proposed system could be employed to solve the under-used public transit problem in any city with a sprawling urban layout.

To obtain the features mentioned above, this chapter is organized as follows for clear illustration: Section 5.1 lists the key components of the ride-sharing module and Section 5.2 introduces the detailed steps for constructing a share-ability graph and algorithm to return efficient assignment in real-time. Section 5.3 proves that the applied algorithm can produce good enough solution within an acceptable time limit and the solution can be further improved over time given enough computational resources. Section 5.4 summaries the main tasks for this module.

5.1. Key Components

Figure 5-1 below summarizes the key inputs, method, and outputs for this module. The input includes the information for the requests, MOD vehicles, and Transit. Request information will include origins, destinations, departure times and expected arrival times for all riders. MOD vehicle information includes the location of each vehicle at any time, the remaining capacity of a vehicle at any time and the undergoing task list. And the transit information includes the stations' location and the schedules for all metro lines.

This information will be collected and inputted into the 4-steps multi-modal ridesharing method, which is illustrated in section 5.2. The method will output the following results: 1) request to vehicle matches; 2) request to request matches for ride-sharing, and; 3) Request to metro schedule matches. Among those, the first two matches will be assigned to vehicles as task lists which contains all the served riders' pickup and drop-off orders.

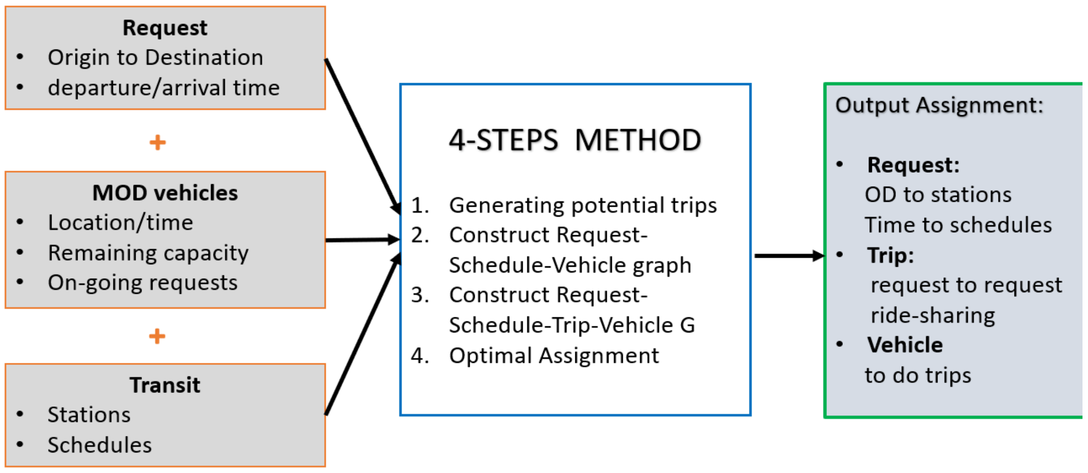


Figure 5- 1. Flowchart of the proposed method

5.2. Problem Statement

Definitions

We consider a fleet size *v* of *m* vehicles with capacity *k,* the maximum number of passengers each vehicle could have at any time point.

A request *r* is consisting of tuple information $\{O_r, D_r, t_r^{start}, t_r^{end}, n_r\}$, indicating its origin O_r, its destination D_r, the time of the request t_r^{start}, the expected time for arrival at destination t_r^{end}, and the number of passengers in the request n_r. A function $\tau(p_1, p_2)$ is defined to compute the travel time from location p_1 to p_2 in space. For this function, when a network map is available, standard techniques can be sufficient to compute the travel time via the shortest path. Considering that this is a typical routing problem which has been extensively explored in the literature, this study will apply an existing tool by directly quoting the google map to get the routing information. However, in theory, this travel time function can be developed using any typical shortest path algorithm or online quotation from third-party routing providers, like google map, Waze, etc.

Using this function output, for any requests, the latest pick-up time t_r^{pl} can be given by

$$t_r^{pl} = t_r^{start} + \Omega \qquad (5\text{-}1)$$

Where Ω denotes for the maximum customer waiting time. And the earliest customer arrival time can be:

$$t_r^{*} = t_r^{start} + \tau(O_r, D_r) \qquad (5\text{-}2)$$

The current state of a MOD vehicle is given by a tuple $\{L_v, R_v, \Delta_v\}$, indicating its current location L_v, the current request list $\{r_1, r_2, ..r_{nv}\}$ the vehicle is serving, and it's capacity Δ_v.

A trip T is a set of requests that can be potentially combined into one vehicle. i.e., $T = \{r_1, r_2, \dots, r_{nT}\}$. A trip may have one or more candidate vehicles for execution.

Transit service capacity is assumed to be sufficiently large to serve all requests at any stations at any time. For a certain request *r*, its entire travel can be decomposed into three sub-trips, as shown in Figure 5-2 below: 1) from the request origin O_r to the origin metro station M_r^o; 2) from the origin metro station M_r^o to its destination metro station M_r^D, and 3) from the destination metro station to the request destination D_r.

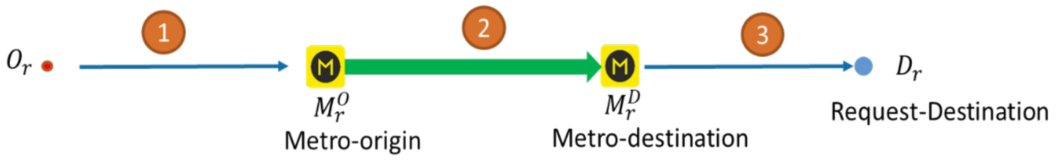


Figure 5- 2. The decomposition of a complete customer request.

The original request tuple can then be represented by 3 tuples correspondingly; the 1st sub-trip is defined by $\{O_r, M_r^O, t_r^{start}, t_r^{metro-start}\}$ where the $t_r^{metro-start}$ denotes the time when the metro departs the origin metro station. The 2nd sub-trip is from origin metro station to destination metro station with tuple $\{M_r^O, M_r^D, t_r^{metro-start}, t_r^{metro-end}\}$ where the $t_r^{metro-end}$ denotes the time when metro line arrives at the destination metro station. The 3rd sub-trip is represented by the tuple $\{M_r^D, D_r, t_r^{metro-end}, t_r^{end}\}$. The optimal trip solution for a rider request has to consist of the three parts mentioned above with integrity.

The earliest arrival time to origin metro station M_r^o is denoted by $t_r^{metro-start*}$, which can be given as:

$$t_r^{metro-start*} = t_r^{start} + \tau(O_r, M_r^O) \tag{5-3}$$

Similarly, the latest arrival time to destination metro station $t_r^{metro-end*}$ is

$$t_r^{metro-end*} = t_r^{end} - \tau(M_r^D, D_r) \tag{5-4}$$

Given the time range ($t_r^{metro-start*}$, $t_r^{metro-end*}$), one can sort out a list of applicable schedules $S = \{s_r^1, s_r^2, \ldots. s_r^n\}$ with each s_r^n corresponding to a 2-element list $\{ t_r^{metro-start.n}$, $t_r^{metro-end.n} \}$. If the returned *S* list is empty, then it means no metro schedule exists to serve the request without violating the arrival time expectation. The system will decline the request and remove it from the unserved request pool, which will be inputted to the next module as part of this module's output from the current iteration.

Problem Formulation

Santi first proposed the share-ability graph concept in 2014 (Santi et al. 2014). They introduced the notion of the share-ability network to model trip sharing in a simple static way and applied classical graph theory methods to assess its potential applicability taking New York City as an example. This concept was later extended by Alonso-Mora in 2017 (Alono-Mora et al., 2017), the authors developed a real-time large-scale ride-sharing framework incorporating the Request-Vehicle (RV) graph construction. They proposed a four-step algorithm, that is constructing an RV graph and then convert it into a Request-Trip-Vehicle graph based on the feasibility check.

They claimed that the algorithm starts from a greedy assignment and improves it via a constrained optimization. Through this approach, their method can quickly return solutions of good quality and converges to the optimal assignment over time if the computational resources are sufficient. The proof of the reactive optimality is presented in section 5.3. This research also employed this concept and designed its own share-ability graph to solve the proposed problem. The following highlights the difference between the previous works and the method proposed by this module:

- Problem difference: The problem stated in this module is different from the typical ride-sharing problem, where request in a typical single mode ride-sharing problem doesn't need to account for the selection of the best feasible metro schedule and ensure the problem integrity among three sub-trips. Those two issues are core challenges presented in this module.
- Methodology difference: To solve the optimal ride-sharing problem in the 1st and 3rd sub-trips with the metro schedule selected in the 2nd sub-trip connecting them, this study designed the Request-Schedule-Vehicle and Request-Schedule-Trip-Vehicle graphs based upon the concept of the share-ability graph. The main reason to use such a graph is to check the trip feasibility and reduce the search space by eliminating the trips violating the basic time window and vehicle capacity constraints. This study adopts such a concept to enhance computational efficiency. However, the share-ability graphs used in the previous work and this

work are fundamentally different. In Alonso-Mora's work in 2017 (Alono-Mora et al., 2017), they constructed one RTV graph and used it to solve the single mode ride-sharing problem. However, this module constructs multiple request-vehicle and schedule share-ability graph to deal with the decomposition of the original request trip.

The detailed methodology will be introduced in the following with a highlight of where the similar concepts are adopted. To ensure the problem integrity, the proposed assignment and routing method is aiming at concurrently optimizing the three sub-trips and consisting of the following steps:

a) Idle Requests Generation: Compute all the possible schedules based on the original request input r using Equations (5-1 to 5-4). A request r_i will be converted into a set of idle requests with each feasible schedule ($r_i^1, r_i^2, ..r_i^n$), where r_i^n stands for idle request r_i taking metro schedule s_n. Note that this study uses the term 'idle request' to denote each feasible request-to-schedule combination, for simplicity purpose. The steps to get a set of feasible schedules for any requests are introduced in the previous section. These idle requests are then inputted into the latter steps, for both the 1st and 3rd sub-trips.

b) Construct RSTV graph for the 1st sub-trip: For each schedule, construct trip candidates for the 1st sub-trip (from origin to metro-station). We firstly create idle requests using the steps in idle request generation then build

RSTV graph for those idle requests. Note that, the RSTV graph not only captures all the feasibility check (waiting time constraint, capacity constraint, etc.) but also makes the multi-stop routing (multi-pickup single drop off) decisions for each trip.

c) Construct RSTV graph for the 3rd sub-trip: For each schedule, construct trip candidates for the 3rd sub-trip (from metro-station to destination). Similarly, we create idle requests from the original request r using the feasible schedules whose end times are the same as the idle request start time and then build RSTV graph.

d) Collect all the RSTV graphs and input them into the Mixed Integer Linear Programming (MIP) to compute the optimal assignment of vehicles to trips. Note that this is the key step to integrate the three sub-trips and the schedule of the metro lines. The 1st and 3rd sub-trips' service availability will reflect on the MIP formulation to ensure the trips' completeness of each request.

The RSTV graphs can be built through the following steps.

a) Computing a pairwise Request-Schedule-Vehicle graph (RSV-Graph). Fed on a set of idle requests $\{r_i^1, r_i^2, ..r_i^n\}$ for a certain original request r_i, the RSV graph is constructed. One may note that each idle request denotes a feasible request-to-schedule combination. In this graph, requests r_i^n, and vehicles v are connected if v can serve r_i^n without violating the defined constraints under the given status of v. And two idle requests r_i^n are connected if they can share a ride. These constraints are typically arrival time constraints, maximum waiting time tolerance, etc.

b) Constructing the Request-Schedule-Trip-Vehicle (RSTV) graph of feasible trips. The trips are the orderly combinations of all the potential ride-shared customers. The trip *T* and a vehicle *v* are connected if *v* can serve the trip within the specified constraints given the present status of vehicle *v*. The constraints include the remaining capacity of the vehicle and time feasibility check for each rider.

Given that solving the batch assignment problem to optimality is NP-hard and computationally expensive, to generate efficient real-time solutions, this study allows a sub-optimal solution to be returned within an allocated runtime budget. However, the goodness of the solution can be improved incrementally up to optimality given sufficient computation time. This property is proved by Alonso-Mora's work in 2017 and will be quoted in section 5.3.

Step a). Generating idle requests

The first step is to convert the original request to a set of feasible request-to-schedule combination, which is called 'idle request' in this study. One request *r* can be represented by a list of requests with all potential metro schedules. For example, the request made by customer *i*, r_i, can be denoted by the tuple $\{r_i^1, r_i^2, ..r_i^n\}$, with each element r_i^n stands for the request *i* taking metro schedule s_n. r_i^n is feasible if the schedule s_n falls in the range of ($t_r^{metro-start*}$, $t_r^{metro-end*}$) for request *i*. The range is generated according to Eq. 5-3 and Eq. 5-4. One thing to note is that for each original request, the idle request lists will be generated for both its 1st and 3rd sub-trips. If either of these two lists is empty, there are no feasible schedules that would match the requested time frame. Thus, the system will reject the request at the end of this iteration.

Step b). Constructing Request-Schedule-Vehicle (RSV) Graph

The first step is to compute the pairwise graph of vehicles and idle requests (RSV-graph), as introduced above. The method needs to determine 1) which idle requests can be pair-wisely combined, and 2) which vehicle can serve which idle requests individually, given their current remaining capacity. This step builds on the idea of share-ability graph originated from Santi's work (Santi et al., 2014).

Two idle requests can be connected in the RSV graph if they can share a ride. This is if a virtual vehicle starting at the origin of one of them could pick-up and drop-off both requests with all constraints satisfied. A cost $\sum_{r=\{1,2\}} (t_r^{end} - t_r^*)$ is associated with each edge $e(r_{i1}^j, r_{i2}^{\ j})$.

A request *r* and a vehicle *v* can be connected if the request can be served by that vehicle given the current status of the vehicle. The status includes the vehicle location, remaining capacity and the requests it is serving at the given time point. The term $travel(v, r_i^j)$ is defined to check if the trip is valid. It will return a binary outcome, where the '1' indicates the vehicle *v* can serve idle request r_i^j without violating the typical time feasibility check or vehicle capacity check. And the returned edges are denoted by $e(v, r_i^j)$. If any of those checks fail, it means that no feasible trip is returned. Note that this function *travel*( ) stands for a set of checks that are designed for presentation simplicity purpose. Those checks typically include time feasibility (e.g., waiting time tolerance, hard arrival time window) and vehicle capacity check.

Figure 5-3 below shows an example of an RV graph. A case with two vehicles and *n* requests is presented. The idle request r_1^1 denotes the original request 1 with the metro schedule s_1. A solid red line is connecting requests when they can share a ride. For instance, r_1^2 and r_3^1 are connected when their combined trip satisfies the time feasibility checks. A dotted black line is connecting a request and a vehicle when the vehicle can serve that request.

Step c). Constructing Request-Schedule-Trip-Vehicle (RSTV) Graph

The second step is to transfer the RSV graph into an RSTV graph by exploring the regions of RSV-graph to find feasible trips. Recall a trip *T* is defined as a set of idle requests $\{r_{i1}^{\ 1}, r_{i1}^{\ 2}, .. r_{ik}^{\ n}\}$. A trip is feasible if the requests it is containing can be picked up and dropped off by a vehicle without violating the defined constraints.

A Request-Schedule-Trip-Vehicle graph $G = (N, E)$ where $N = (R, T, V)$ is then presented. For the edge set E, an idle request $r_i^j \in R$ and a trip $T \in \mathrm{T}$ is connected if $r \in T$ and T is a valid trip.

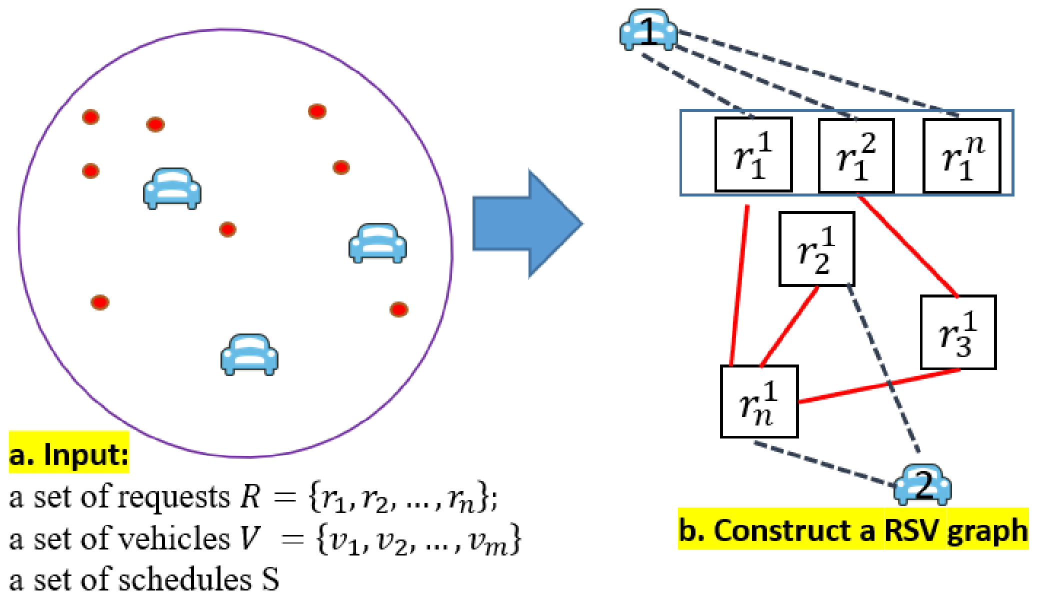

a. Input:
a set of requests $R = \{r_1, r_2, \ldots, r_n\}$;
a set of vehicles $V = \{v_1, v_2, \ldots, v_m\}$
a set of schedules S
1
r_1^1
r_1^2
r_1^n
r_2^1
r_3^1
r_n^1
2
b. Construct a RSV graph

Figure 5- 3. An example of a pairwise RSV graph

Figure 5-4 shows an example of an RTV graph, a trip $T \in \mathrm{T}$, and a vehicle $v \in V$ are connected when the vehicle can serve the trip, i.e., $travel(v, T)$ satisfy all the time constraint and capacity constraint check. $travel(v, T)$ is a function that returns a binary value indicating whether this vehicle could server the trip T while satisfying all the feasibility checks. Each trip is an orderly sequence of no less than one idle request. This function will check the feasibility over all possible combinations when the trip consisting of no more than three idle requests. Due to the computational power limitation, some heuristics can be developed to speed up the process without enumeration. This study sets the maximum passenger capacity of a MOD vehicle to be three and conducts the feasibility check for all combinations. Also, with each edge $e(T, v)$, the cost C of connecting the trip and pick-up is stored.

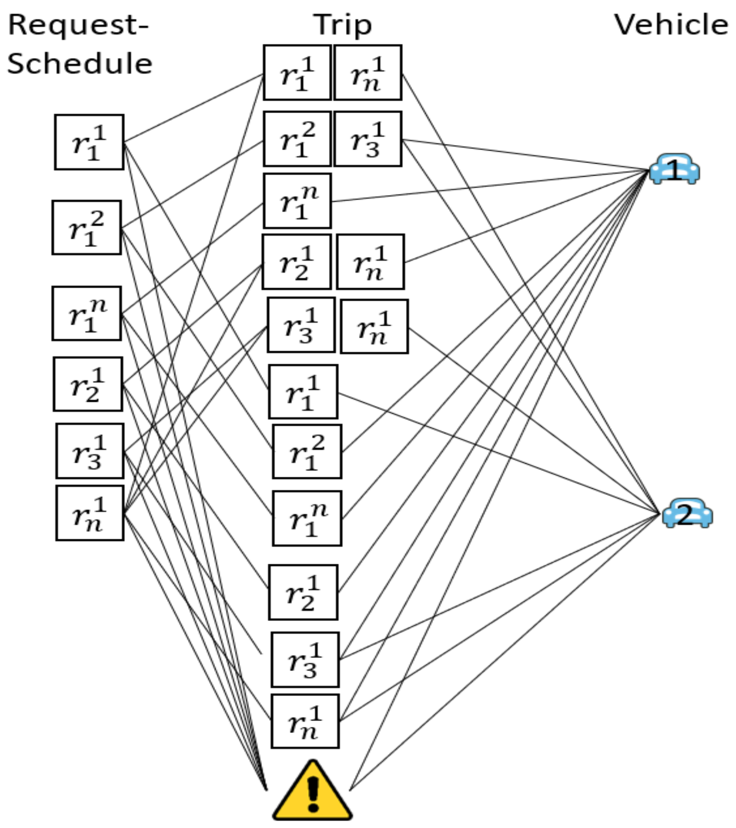


Figure 5- 4**.** An example of an RSTV graph

<u>Step d). Solving MIP for optimal assignments of trips to vehicles</u>

The last step is solving the optimal matches of vehicles and trips based on a combination of all RSTV graphs, given all the constructed RSTV graph in the current iteration. This is formulated as a mixed integer linear programming. Table 5-1 below lists the key notations for the MIP.

Table 5- 1. Key notations for optimal assignment MIP model.

Notation	
Request	
r	A request r is defined by five tuples $(O_r, D_r, t_r^{start}, t_r^{end}, t_r^*)$
S_r	Source location
D_r	Destination location
t_r^{start}	Start time
t_r^{end}	End time
t_r^*	Earliest possible arrival time $(t_r^* = t_r^{start} + \tau(O_r, D_r))$
$\tau\ (q_1, q_2)$	A function to compute the travel time between location $q_1$and q_2.
Vehicle	
v	The current state of v is defined by three tuples (L_v, R_v, Δ_v)
L_r	Vehicle location
$R_v = \{r_1, r_2, \dots, r_{nv}\}$	The list of requests, the v is currently serving
Δ_v	Vehicle capacity
Schedules	
S	A set of schedules $S = \{s_1, s_2 \dots s_n\}$ that fits within the earliest departure time at origin metro station and the latest arrival time at destination metro schedule.
Trip	
T	A set of requests that can be combined and served by a single vehicle, i.e., $T\ = \{r_1, r_2, \dots, r_{nT}\}$. A trip may have one or more candidate vehicles for execution.

Variables include the following:

- $\varepsilon_{i,j,s}^{O \to M^o}$: A binary variable for each edge $\mathrm{e}(\mathrm{T}_i,\mathrm{v}_j)$ between a trip $\mathrm{T}_i^{O \to M^o} \in \mathrm{T}^{O \to M^o}\ and\ \mathrm{V}_j \in V^{O \to M^o}\ for$ schdule s . It is '1'when the trip can be served by the vehicle. Otherwise, it will be equal to zero;
- $\varepsilon_{i,j,s}^{M^D \to D}$: Similar to $\varepsilon_{i,j,s}^{O \to M^o}$, if equals 1 means a vehicle v_j is assigned to trip T_i using metro schedule s;

- $\chi_k^{O\to M^o}$: a binary variable, equals 1 if a request *k* is not served by any vehicles for 1st sub-trip from O to M^o ;
- $\chi_k^{M^D\to D}$: A binary variable, equals 1 if a request *k* is not served by any vehicles for 3rd sub-trip from $M^D to\ D$;
- $c_{i,j,s}^{O\to M^O}$: $=\sum_{r\in T_i}(t_r^{end}-t_r^*)$ is the sum of the time difference between the expected arrival time to the metro station M^O and the earliest arrival time. This term is returned by $travel(v_j,T_i)$;
- c_{ko} : A large enough positive constant to penalize the unserved requests.

The objective function is specified below with the goal to maximize the number of served requests while minimizing the cost of all trips. The 1st and the 3rd sub-trips involving MOD ride-sharing will be combined and to be solved using the same MIP formulation.

$$\text{Min } C(\chi)=\sum_{i,j,s}c_{i,j,s}^{O\to M^O}\varepsilon_{i,j,s}^{O\to M^O}+\sum_k c_{ko}\chi_k^{O\to M^O} \tag{5-4}$$

The first term $\sum_{i,j,s}c_{i,j,s}^{O\to M^O}\varepsilon_{i,j,s}^{O\to M^O}$ is the cost of delay between the expected arrival time to the origin metro station and the earliest arrival time to it. This helps to minimize the customer in-metro station waiting time. Note that besides this objective, one can also change this term to reflect the operational cost or profitability in order to present other optimization concerns. To be more specific, the term $c_{i,j,s}^{O\to M^O}$ can have other forms rather than the arrival time delays. For example, if one aims to minimize the cost of the operator, it can be the monetary cost for serving each rider request. And in the RSTV graph, the cost of each edge will also need to be the operating cost accordingly. The second term $\sum_k c_{ko} \chi_k^{O \to M^O}$ is the penalty for all the unserved vehicles. Note that in this study, an unserved request can only happen when it fails the time feasibility check at some iterations.

The constraints are presented in the following Eq. (5-5~5-7):

1) Each vehicle is assigned to at most one trip:

$$\sum_{s \in S} \sum_{i \in I} \varepsilon_{i,j,s}^{O \to M^O} \leq 1 \tag{5-5}$$

2) Each request is assigned to each vehicle or ignored:

$$\sum_{s \in S} \sum_{i \in I} \varepsilon_{i,j,s}^{O \to M^O} + \chi_k^{O \to M^O} = 1 \tag{5-6}$$

3) Each request is served by the same schedule s for both its 1st sub-trip ($O \to M^O$) and its 3rd sub-trip ($M^D \to D$):

$$\sum_{s \in S} \sum_{i \in I} \varepsilon_{i,j,s}^{O \to M^O} - \sum_{s \in S} \sum_{i \in I} \varepsilon_{i,j,s}^{M^D \to D} = 0 \tag{5-7}$$

The graphical presentation of the optimal assignments is shown in Figure 5-5. The bold red line denotes the generated optimal request to trip and trip to vehicle matches. With the trip definition, the selected optimal trip is the best request-to-request matches with a selection of metro schedules concurrently.

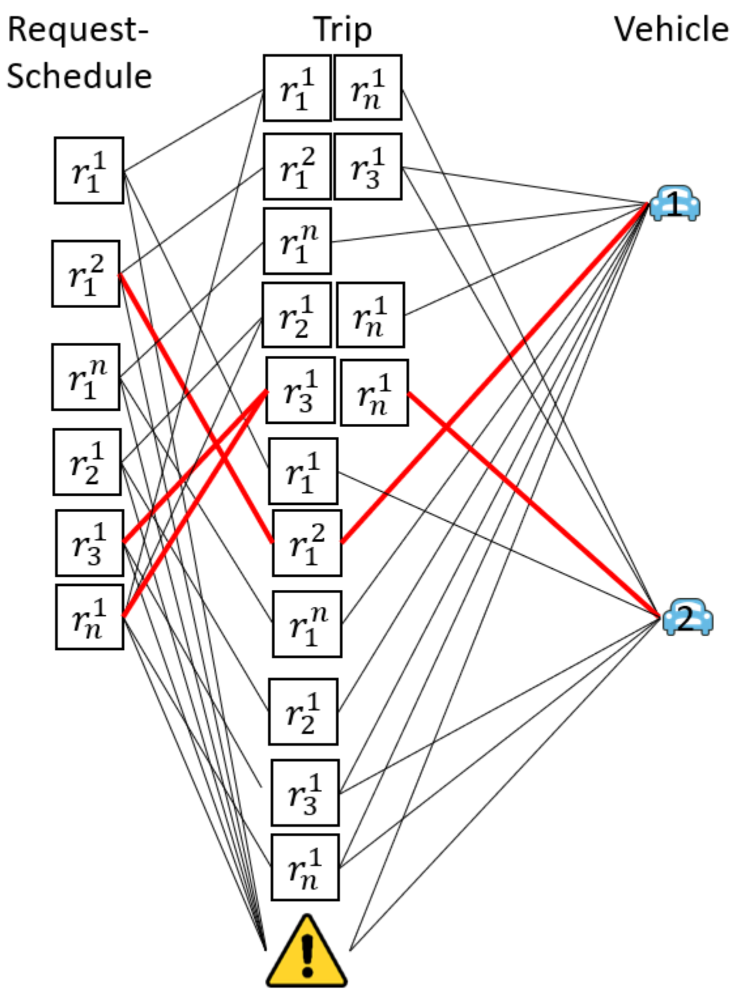


Figure 5- 5. A graphical example of an optimal assignment

5.3. Theoretical Basis

Since this study adopts the concept of a share-ability graph, which has been used in the previous literature. The complexity and optimality conditions are determined by the structure of the graphs which are specifically both bipartite graphs. This section quotes the theoretical exploration of the ride-sharing method, including the complexity of the problem and the optimality statement inspired by Alonso-Mora's work in 2017.

Complexity

The number of binary variables in the MIP is the sum of the number of edges $e(T, v)$ in the RSTV graph and the total number of requests in any iteration. So the worst case scenario in theory without all previous checks are presented in the following equation when all the trip combinations are explored:

$$m \frac{n!}{v!(n-v)!} \sim O(mn^{v}) \tag{5-8}$$

Where *m* is the total number of vehicles, *n* is the total number of idle requests, and *v* is the maximum capacity of all vehicles. This is basically an exhaustive search of all the possible combinations and reaches the feasible one at the last combination. The worst-case scenario can only be reached if no map-partitioning is done and all the trip combinations are possible, and all the vehicles can serve all the requests. This, however, is not true in our problem because the entire map is partitioned into smaller zones in module one and then a serial time feasibility check is performed while constructing the edges of RSTV graph. The worst-case scenario of this problem is improved compared with the one in equation 5-8.

Optimality Claim

The following statement is quoted from a similar work (Alonso-Mora et al., 2017).

"(Optimality). The method is optimal, given enough computational time."

If not considering computation efficiency, $travel(v,T)$ can do an exhaustive search for any particular vehicle to trip assignment. Under the scenario that the RSV and RSTV graphs are both complete graphs, all possible trips with less than the maximum vehicle capacity will be explored. Moreover, all valid trip combinations are added to the RSTV graph. That is to say, under an exhaustive search, this algorithm can solve the problem to optimality.

The MIP formulation contains all the possible trip-vehicle assignments. Once given enough time, the optimal solution can be guaranteed through the branch-and-bound method. However, it may lead to an exhaustive search with very expensive computational time. Nevertheless, the advantage of the method is that, through decoupling, reducing the RSV and RSTV graph size, and the separation of valid routes and trips, good solutions can be found efficiently.

5.4. Summary

This chapter presented the steps used to construct the RSV and RSTV graphs. A MIP formulation is then applied to generate the optimal trip to vehicle assignments with an objective of minimizing total cost for served requests together with the penalties for the unserved requests. Note that this objective function can apply to other optimization concerns by changing the definition of the edge cost in RSV/RSTV graph. For instance, one can define the edge cost as the profit gain from serving each request, and the objective function will be formulated to maximize the profit of the overall system. Besides the steps mentioned above, several points need to be highlighted:

- Travel time feasibility check

This problem occurs when the complete travel request is decomposed into 3 sub-trips. The integration problem will occur if the method is not properly designed to capture their interactions. For example, the 2nd in-metro sub-trip is reflected by the request-schedule list and inputted into the RSV and RSTV graph computing. To ensure the system can serve the customers, the time feasibility check is done in the following ways: 1) usage of the function $travel()$ to check if the vehicle-to-trip/ vehicle-to-request assignment is feasible; 2) through potential metro schedule list. If empty, it means it is impossible to serve the request; 3) Solving the MIP, if the solution does not return the request assignment, then it will remain in the queued request pool and re-enter the next iteration until it is either served or declined.

- Vehicle Capacity

The maximum vehicle capacity is fixed. Although the RSV graph is a pairwise share-ability graph, the proposed ride-sharing method is not limited to two ride-sharing requests. This is because a vehicle with remaining capacity will still be counted as available and the onboard request will be matched with the queued requests. If more than two requests can share a ride, the system will still match them until the maximum vehicle capacity is reached.

Chapter 6: Module-3 Rebalancing

Due to the fundamental imbalance of the customer pick-up and drop-off in any car-sharing or ridesharing system that downgrades the system performance, an efficient fleet management strategy needs to be designed. The vehicles far away from the area of the current request may be idle and unable to serve any requests. At the same time, requests may lack vehicles for service if no available ones are nearby to satisfy their hard time windows. In a MOD system, this can be unacceptable for both the idle vehicle drivers and the customers. Learning from such an unpleasant experience, the driver may not be willing to provide an on-demand service even when supplies are highly in need. Moreover, the customers will walk away from the service and thus limit its benefit in the long run.

For areas with high rider demands, it is likely that more requests can occur in the same area where the services vehicle numbers are not sufficient to serve all of them. Similarly, areas with relatively lower requests may have residual MOD vehicle supply. Under such assumptions, the main tasks for the rebalancing module are:

a) Prediction of the future demand and relocation of the vehicles in preparation for the incoming demand;
b) Generation of the optimal relocation plan to move vehicles most efficiently.

Section 6.1 presents the framework of this module with the highlights of the key inputs and outputs. Section 6.2 illustrates the detailed steps to develop a reliable prediction model at scale by comparing the performance of three advanced demand forecasting models. Section 6.3 develops the optimization model which takes as input the forecasting results and generates the optimal vehicle relocation plan. Finally, Section 6.4 summarizes the main tasks of this module along with its contribution.

6.1. Proposed Framework

Figure 6-1 below presents the framework of the rebalancing module. The key input includes the information of the status for MOD vehicles and demand patterns. For MOD vehicles, they can either be assigned to a certain station or en route from one station to another. The demand input contains three categories. They are: 1) the queued requests in the past, 2) the number of requests at present, and 3) the demand forecast in the next look-ahead time window. Besides the first two categories that are already recorded in the request information pool, the last part, which represents the future demand for pickup and dropoff, is unknown. So one of the main tasks of this module is to develop an accurate prediction module that could also deliver reliable forecasting within the required time frame. Three models were developed and compared in terms of both computational efficiency and the prediction accuracy aspects. The one with preferable performance in both aspects is selected. This demand forecasting results, combined with the existing request information will be inputted into a mixed integer programming model with an objective of minimizing total vehicle time spent for the rebalancing task to generate the best fleet relocation plan. The outputted rebalancing plans will later be inputted to update the vehicle information pool and used as the input for the next iteration.

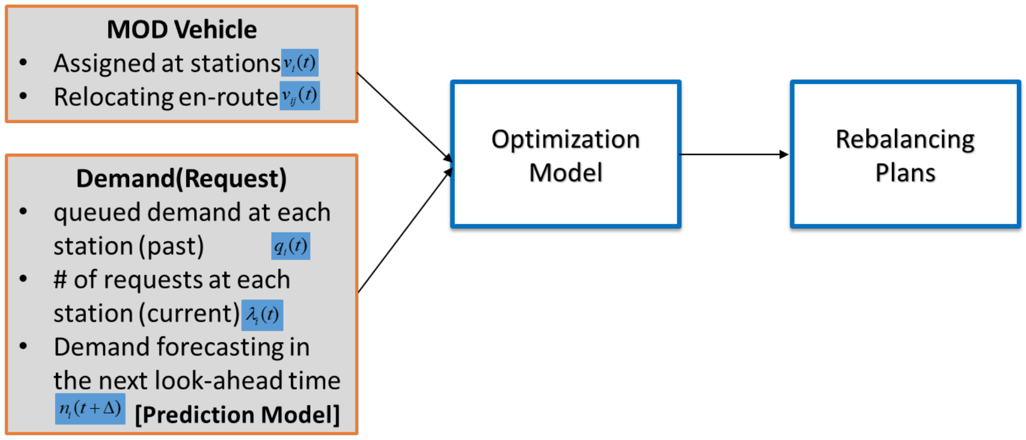

MOD Vehicle
Assigned at stations $v_i(t)$
Relocating en-route $v_{ij}(t)$
Demand(Request)
queued demand at each station (past) $q_i(t)$
# of requests at each station (current) $\lambda_i(t)$
Demand forecasting in the next look-ahead time $n_i(t+\Delta)$ [Prediction Model]
Optimization Model
Rebalancing Plans

Figure 6- 1. Flowchart of the rebalancing module

6.2. Demand Forecasting Model

As one of the key inputs, the demand forecast $n_i^{desired}(t+\Delta)$ needs to be sufficiently accurate to guarantee an efficient rebalancing plan. It needs to concurrently reflect the queued demand $q_i(t)$, current demand $\lambda_i(t)$, and future demand $\lambda_i(t+\Delta)$. Among all three components, the queued demand and the current demand are recorded in the request information pool. The task left is to predict future demand with the necessary precision. This section presents the general steps in exploring the appreciate models and the key concerns to develop a demand prediction model, especially when using real-world demand data. Although the performance of the proposed models is dependent on the dataset of usage, the steps and key considerations are generic and can be adapted to be applied to other prediction tasks with similar requirements. To better fit the real-world MOD request pattern, this section applies DC taxi open data as the source for model development. A desirable prediction model should bear the following features:

- Deliver accurate prediction for the station-based pickup and drop-off demands in the next look-ahead window;
- Generate prediction results in a timely manner, as the model will be trained in real-time with limited input data to satisfy computational efficiency requirements

The detailed steps and concerns are discussed in the following section.

Data Source

This study applies the open taxi data published by the District of Columbia Department of For-Hire Vehicles for the modeling. The dataset contains the taxi trip records whose origin and destination are located within the geological boundary of DC. Each record contains the latitude, longitude and times for pickup and drop-offs. To align with the purpose of the module, all other irrelevant information, such as trip cost, payment type, etc. are removed from the modeling dataset.

A sampling of taxicab data from July 25 to July 27, 2017, which consists of three consecutive workdays, is used. After filtering out the trips without necessary information (coordinates and time for pickup and drop-offs), a total of 28611 records are kept for analyzing. To avoid the impact of special event, the selected days are not near any holidays. Also, the input data is incorporated into the system which requires the prediction model to be trained in real-time. So this model needs to achieve the desired level of precision with limited input data. This study trains the model on the previous two days' data to predict the following one day. In the real application, this forecasting model can work in a moving window mechanism, using the most recent 2 weekdays' historical demands to predict the next weekday. Note that the primary use case of the proposed MOD system is to serve as a valid transportation mode for daily commuters during weekdays to alleviate urban congestions in the long run.

Note that the prediction would exclude the weekend demand pattern from the analysis. This is because the trip purpose is inherently different between weekday and weekend. Also, the optimal input data size should be determined to achieve a balance between computational expense and prediction precision. The longer the input historical serial data is, the better it could, intuitively, predict the future demand. However, on the other hand, this makes online training less computationally efficient. The optimal input size should always be determined on a case by case basis, based on the professional judgment of the system planner. This study inputs data from two days to predict the next day, as an example. However, it is acknowledged that more work can be done to enhance the prediction accuracy. For example, testing on the seasonality effect by including a longer historical input. Regarding that the demand can often have a weekly repeatable pattern, future work can examine the best input window size by testing on past weekly inputs. However, the ultimate goal of this section is to present the general steps and key concerns that one should bear in mind when applying this framework. So building a state-of-art forecasting model according to this particular dataset is not a priority here.

Under the scope of this dissertation, taxi dataset will be used in the following two ways:

- <u>Development of a demand forecasting model</u>: The data contains the trip pickup and drop-off time information with one-hour precision. So the station-based demand will be forecasted on an hourly basis.
- <u>Simulation Experiments</u>: The data will be directly used as the request input to the numerical analysis case-2. All performance measures will be summarized based on this real-world taxi demand.

<u>Demand Evolution</u>

Figure 6-2 below plots the demand evolution against time. The coordinates of pickup and drop-off points were mapped on the geo-layout of Washington D.C. Due to the highly similar hourly demand pattern distribution among the selected three days, this section picked one day for presentation purpose. Note that the pickup and drop-off points are denoted as black and red, respectively. The darker the color is, the higher the demand in that region.

As displayed, the demand is showing two peaks. One is the morning peak (8:00 AM), where most of the drop-off points are located in the center of DC. The other peak is afternoon peak (5:00 PM), where most of the pickups are located at the central business zone while the red drop-off dots are located towards the outside. This is a typical workday traffic pattern, where the commuters travel from outside to CBD to go to work and reverse the pattern from CBD to outside in the afternoon peak hours.

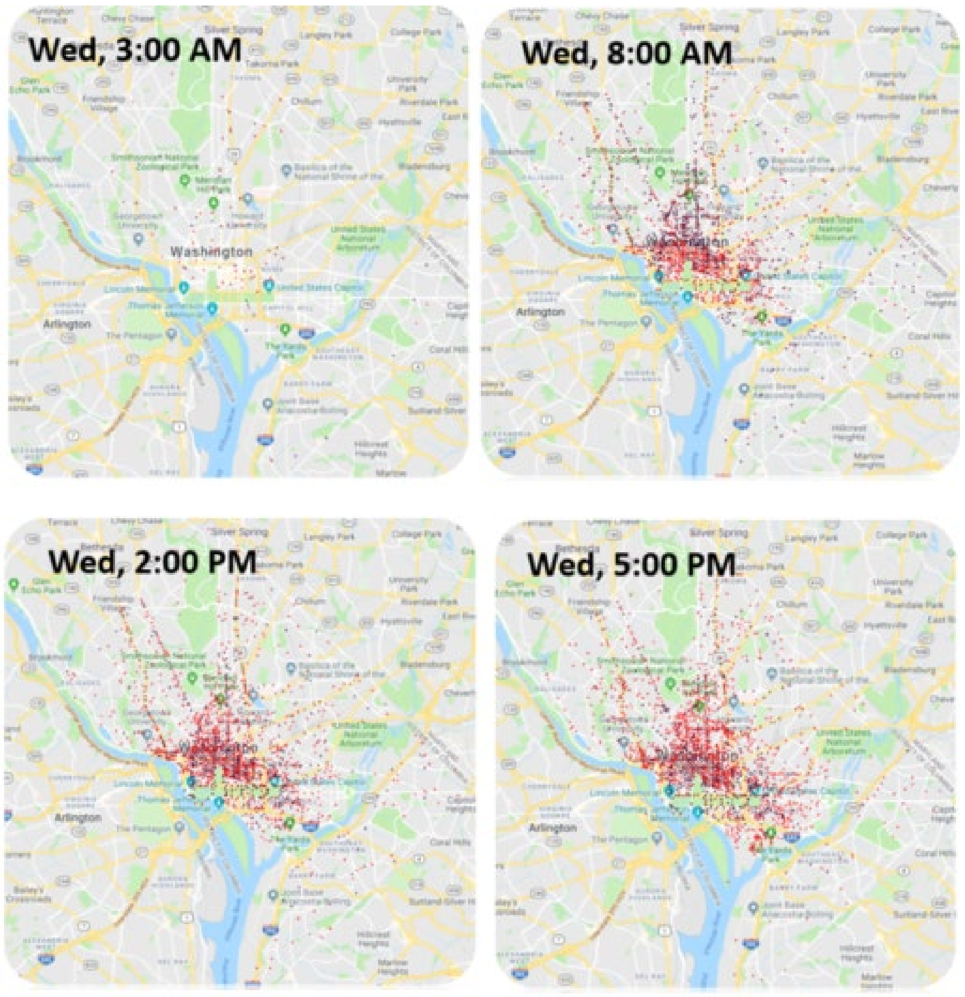


Figure 6- 2. Hourly demand pattern visualization

Hourly Station-based Demands

In the pre-processing step, the original taxi trips were assigned to each station according to their pickup and drop-off coordinates. This was done by executing the module-1 map-partitioning algorithm to get the station-based demand. Figure 6-3 shows the selection of nine stations. They were chosen based on the map-partitioning results. Reasonably well clustering results should bear a balance between the within-cluster sparsity and between-cluster discrepancy. In this case, taking these nine stations would allow the demand points assigned to the same station located nearby each other, which represents the within-group sparsity. Meanwhile, the demand points assigned to different stations are relatively more apart, which represents the between-group discrepancy. Note that any trip with its origin and destination assigned to the same station is illegitimate and is removed from the later analysis.

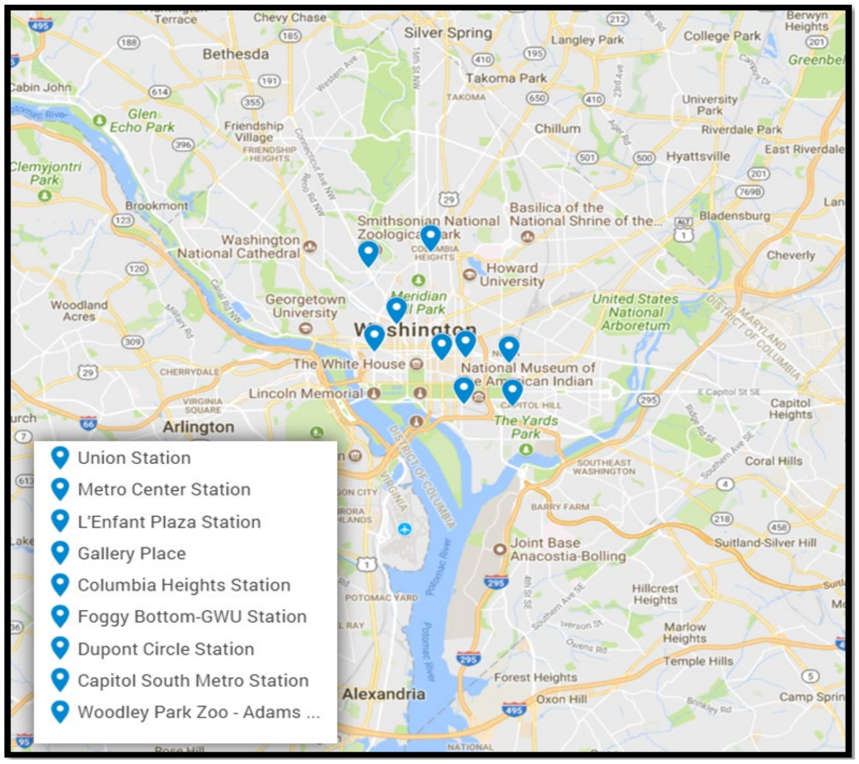
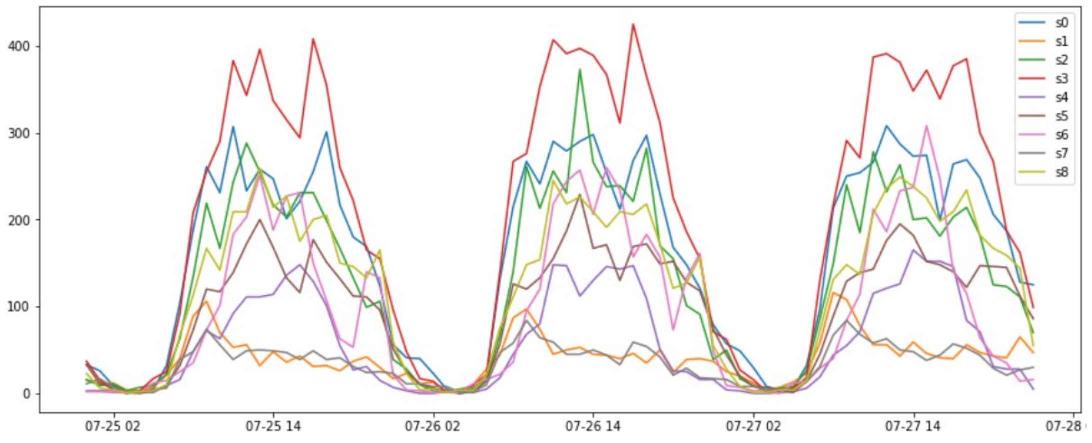
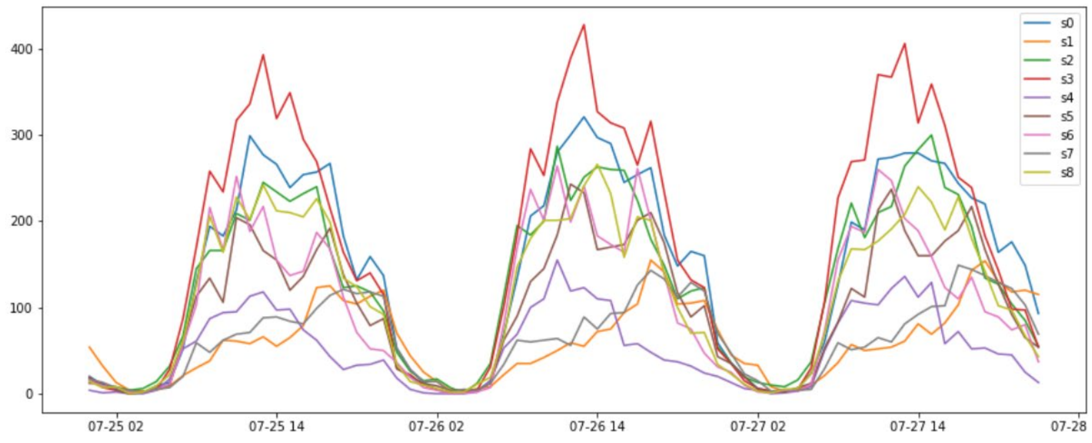


Figure 6- 3. Location of the selected nine stations

Figure 6-4 plots the hourly station-based demand for three consecutive days. Lines with different colors denote the different stations. From the plot, one can see that there is a noticeable difference among stations, but the daily pattern for the same station is repeatable, as it generally forms a daily bell-shape.



(a) Hourly station-based pickups



(b) Hourly station-based drop-offs

Figure 6- 4. Station-based hourly demand for three consecutive days

Model Building

To select the best model fitting the purpose of rebalancing module, three models were trained and compared. The one with the desired precision and scalability is selected. The three models are a base model, which is the most popular time series prediction model widely applied in many fields and two advanced deep learning models, applied due to their well-known prediction power. They are:

- Base Model: ARIMA
- Deep Neural Networks -1: Recurrent Neural Networks (LSTM)
- Deep Neural Networks -2: Convolutional Neural Networks

The first two days are used to train the model and the last day is used for testing purposes. The following summarizes the data split:

- Training set: July 25th, 2017 00:00 am ~July 26th, 2017 24:00 pm
- Testing set: July 27th, 2017 00:00 am ~ 24:00 pm

Base Model: ARIMA

The autoregressive integrated moving average (ARIMA) model is a generalization of mixed autoregressive moving average (ARMA) model. The autoregressive part of ARMA indicates the evolving variables are regressed on their own lagged values. The moving average part of ARMA indicates that its component error is a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The general form of the ARMA model is shown below:

$$y_t = \mu + \varphi_1 y_{t-1} + \ldots + \varphi_p y_{t-p} - \theta_1 e_{t-1} - \ldots \theta_1 e_{t-q} \tag{6-1}$$

where θ_p are the moving average parameters, φ_p are the autoregressive parameters, μ is a constant and *e* denotes the error term. The model is generally denoted as ARMA (*p,q*) where *p* is the number of autoregressive parameters in the model and *q* is the moving average parameter. A time series $\{y_t\}$ has an integrated autoregressive-moving average model. The d^{th} difference $X_t = (1-B)^d Y_t$ is a stationary ARMA process, where B denotes the back shift operator. That is, $BY_t = Y_{t-1}$. We say Y_t is a ARIMA process (*p,d,q*) and X_t follows a ARMA process (*p,q*).

Before model training, a stationarity check has to be performed to determine whether differencing is needed. To do so, the Augmented Dickey-Fuller test was performed. The null hypothesis is that a unit root is present in the autoregressive model and the alternative hypothesis is that there is no unit root or the data is stationary. Besides the stationarity hypothesis testing, the rolling mean and standard deviation are also plotted (as shown in Figure 6-5 as an instance) and they are showing a flat trend during the three-day sampling time. As a stationary process has the property that its mean and variance do not change overtime. This again proves the series' stationarity.

Table 6-1 below shows one can reject the null hypothesis as all the p-values are less than 0.05. So the stationarity assumption holds for the pickup and drop-off series for all stations. Since the original data are stationary, differencing is not needed. Hence, *d* is set to be 0.

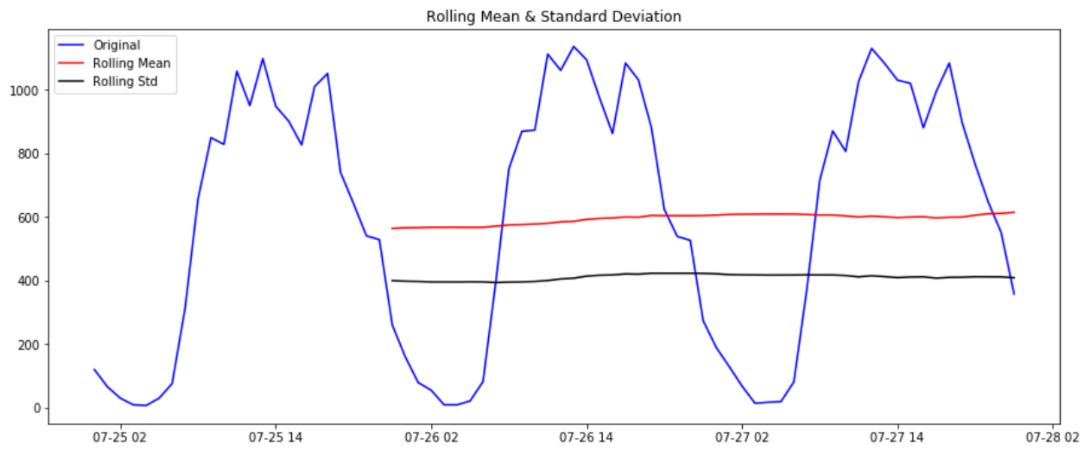


Figure 6- 5. Rolling mean and standard devation plot for station 3

For each station, the hype-parameters are tuned to get the smallest AIC, where the lower the AIC, the better the model fits the data. Table 6-2 summarizes the best combination of *(p,d,q)* for each series as well as the Root-Mean-Square-Error for both training and test datasets.

Table 6- 1. Results for stationarity assumption tests

	Pickup		Drop-off	
Station index	Test statistic	P-value	Test statistic	P-value
1.000	-4.640	0.000	-5.440	0.000
2.000	-4.118	0.001	-4.908	0.000
3.000	-5.862	0.000	-5.260	0.000
4.000	-5.444	0.000	-4.753	0.000
5.000	-5.315	0.000	-3.597	0.006
6.000	-4.862	0.000	-3.263	0.017
7.000	-4.603	0.000	-5.513	0.000
8.000	-4.373	0.000	-5.518	0.000
9.000	-3.865	0.002	-5.640	0.000

Table 6- 2. Results summary for ARIMA models

Station index	Pickup				Drop-off			
	(p,d,q)	AIC	Train RMSE	Test RMSE	(p,d,q)	AIC	Train RMSE	Test RMSE
1	(2, 0, 1)	493.88	40.15	43.88	(5, 0, 1)	467.34	30.77	26.43
2	(2, 0, 1)	398.22	13.92	26.53	(5, 0, 3)	404.2	12.74	29.25
3	(5, 0, 1)	498.7	38.72	43.2	(4, 0, 1)	465.01	29.66	34.6
4	(2, 0, 2)	510.2	47.61	35.82	(2, 0, 2)	492.92	40.31	45.82
5	(4, 0, 0)	419.7	17.95	34.92	(3, 0, 1)	395.3	14.23	23.5
6	(5, 0, 1)	450.28	23.03	34.36	(5, 0, 1)	468.12	28.44	31.22
7	(4, 0, 1)	500.44	39.19	21.55	(4, 0, 1)	502.45	39.85	41.09
8	(2, 0, 1)	379.08	11.5	17.79	(2, 0, 1)	409.4	16.19	20.17
9	(2, 0, 3)	467.83	29.28	25.36	(2, 0, 2)	452.04	27.05	20.72

For each serial data (pickup or drop-off demand), its RMSE for training and testing are similar to each other. This indicates that the ARIMA models perform well with no under-fitting nor overfitting. To furthur evaluate the performance of the ARIMA models, this study compares its root mean square error with the simple historical average. Table 6-3 below summarizes the comparison results over the testing day. For simple average, the predicted demand for a certain hour is the historical demand average of same hour.

From the table below, we can clearly see that the ARIMA model outperform the simple historical average on every stationary pickup and drop-offs, which proves that the ARIMA model can serve as a good reference base that outperforms the naïve simple average. The following two models will be furthur compared with this base in terms of prediction accuracy and computational expenses.

Table 6- 3. Comparison between ARIMA and historical average on testing set

Station index	Pick-up			Drop-off		
	Simple Ave. RMSE	ARIMA RMSE	Redu. RMSE	Simple Ave. RMSE	ARIMA RMSE	Redu. RMSE
1	62.71	43.88	18.83	35.11	30.77	8.68
2	40.88	26.53	14.35	44.23	12.74	14.98
3	62.18	43.2	18.98	48.40	29.66	13.80
4	48.77	35.82	12.95	71.44	40.31	25.62
5	57.58	34.92	22.66	35.54	14.23	12.04
6	52.77	34.36	18.41	48.68	28.44	17.46
7	40.83	21.55	19.28	50.53	39.85	9.44
8	28.47	17.79	10.68	28.31	16.19	8.14
9	35.16	25.36	9.80	27.13	27.05	6.41

Note:
**Redu. RMSE: denotes the reduced RMSE difference between ARIMA and Simple Historical Average.*

Deep Neural Networks-1: Recurrent Neural Networks

This study further adopts deep leaning techniques to perform demand forecasting due to their widely known prediction power. Recurrent neural networks (RNNs) is a class of artificial neural networks where connections between nodes form a directed sequence. Unlike conventional feedforward neural networks, RNNs can use their internal states, which also are called memory, to process the sequential inputs. This technique has been widely applied to the area of speech recognition, and connected handwriting recognition and has achieved promising results. Similar to conventional feed-forward neural networks, RNN applies a gradient descent algorithm to iteratively minimize the error term between predicted and observed values of the training data. Then it adjusts the weights of its input to minimize the prediction error.

However, by doing this, there are two major obstacles, one is the exploding gradients, and this would result in the search jumps too far that could miss a local optimal region and get trapped in an area without improving the cost function.

Another problem is the vanished gradient; this happens when the change in gradient is too small. Thus it either takes a long way to reach a local optimal point, or the learning process stops when it reaches the maximum iteration limits. This is extremely severe when it comes to training data with a long sequence, for example, time series data with a yearly or seasonal pattern.

As a special type of RNN, Long short-term memory recurrent neural networks (LSTM) was first proposed by Hochreiter and Schmidhuber(1997) to avoid the vanishing gradient problem faced by conventional RNN. It is normally augmented by recurrent gates called 'forget' gates. Simply stated, LSTM networks prevent back-propagated errors from vanishing or exploding by learning when to remember and when to forget. This all done by controlling the change in the gradient through the collaboration among its three gates: input, forget and output gates. These gates determine whether to let new input in (input gate), delete the information because it isn't important (forget gate) or to let it impact the output at the current time step (output gate). Figure 6-6 provides the graphical illustration of LSTM.

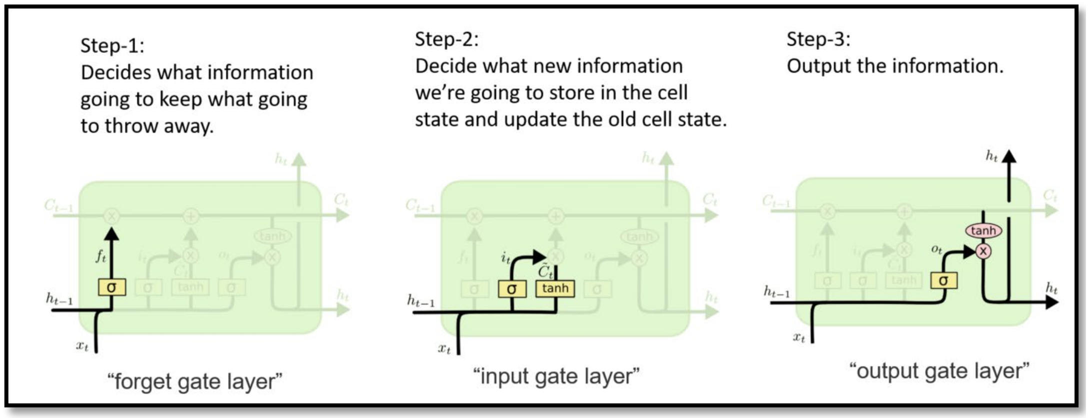

Step-1:
Decides what information going to keep what going to throw away.
Step-2:
Decide what new information we're going to store in the cell state and update the old cell state.
Step-3:
Output the information.
"forget gate layer"
"input gate layer"
"output gate layer"

Figure 6- 6. Graphical presentation of a typical internal state of LSTM

This study also trained the LSTM for each station-based hourly demand. Similar to all other statistical models, deep neural nets also require the modeler to select the best combination of a set of hyperparameters for their best performance.

They are:

- Number of Epochs: Total number of forward and backward passes for all training data. One epoch refers to one forward and backward pass.
- Batch size: Number of training samples in one epoch. The higher the batch size, the more memory space it requires.
- No of neurons: Number of LSTM neurons in the training network.

Table 6-4 summarizes the hyperparameter tuning results, which include the batch size, number of epochs and the number of neurons.

Table 6- 4. Hyperparameter results for LSTM models

Station Index	Pickup			Drop-off		
	Batch size	No. epoch	No. neurons	Batch size	No. epoch	No. neurons
1	1	500	10	1	500	10
2	1	500	9	1	500	10
3	1	500	9	1	500	10
4	1	600	12	1	500	10
5	1	500	9	1	500	10
6	1	500	9	1	500	10
7	1	500	10	1	500	10
8	1	500	10	1	500	10
9	1	1000	10	1	500	12

Deep Neural Networks-2: Convolutional Neural Networks

Despite the LSTM's reputation in sequential data forecasting performance, the rebalancing module also requires the model to be computationally efficient while maintaining adequate prediction accuracy. Because of this, this study further applies another type of deep learning technique, convolutional neural networks (CNN), to test whether it could be a better fit for online training purposes with a comparable prediction accuracy.

Unlike LSTM which takes into account all past observations to derive the prediction for the future, CNN puts more emphasis on the most recent past observation by assuming that they would be more closely related to the next time observation than past experiences that are far before it. Therefore, the CNNs are better in finding local patterns with less past data feed. This type of deep neural networks is widely applied in image processing and speech recognition field with revolutionary advancement in its real-world applications. CNN reduces the dimension of input data by an operation named ‘pooling,’ which is mapping the original input date through a convolutional filter to produce a feature map and then apply conventional neuron network operations. By doing this, CNN can effectively shorten the training time and combat the typical overfitting issues that commonly occur with deep neural networks.

Similar to the hyper-parameters tuned for LSTM, the CNNs also are pre-trained to select the best value sets. Note that all the station-based pickup and drop-offs share the same hyper-parameter results, which is batch size=2, No. of epochs= 400. However, different from the LSTM that requires the 2-day ahead inputs; CNNs reduce the input size to only 1 day ahead, which is called the window size in CNN. The input size is changed because CNN usually requires less input compared with LSTM in sequential prediction.

Model Comparison

To compare the three models’ performance and select the one that fits the ideal model feature, this study selected two comparison measures. One is the root mean square error (RMSE) as defined in the equation below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_{observed,i} - y_{predicted,i})^2}{N}} \tag{6-2}$$

where N is the total number of observations. Another comparison basis is the Mean Absolute Error (MAE) defined in the Equation 6-6 below:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_{observed,i} - y_{predicted,i}\right| \tag{6-3}$$

Table 6-5 and Table 6-6 below summarize the RMSE and MAE for all model's performance, respectively. There are columns that summarize the reduction in both error terms when comparing two deep learning models to the based model, ARIMA. One can see that both LSTM and CNN significantly outperform the ARIMA model. LSTM reduced the RMSE compared with base ARIMA model ranging from 16% to 69% and reduced MAE from 13% to 67%.

Table 6- 5. Summary for RMSE for proposed models

	Pickup					Drop-off				
Station index	ARI-MA	CNN	RNN	Redu. CNN-ARIMA	Redu. RNN-ARIMA	ARI-MA	CNN	RNN	Redu. CNN-ARIMA	Redu. RNN-ARIMA
1	43.88	23.19	20.81	47%	53%	26.43	19.93	19.07	25%	28%
2	26.53	16.65	12.58	37%	53%	29.25	16.76	14.5	43%	50%
3	43.2	35.24	26.59	18%	38%	34.6	27.52	23.11	20%	33%
4	35.82	26.19	22.43	27%	37%	45.82	35.73	31.62	22%	31%
5	54.92	20.26	16.92	63%	69%	23.5	19.25	16.27	18%	31%
6	34.36	20.36	16.67	41%	51%	31.22	26.39	22.93	15%	27%
7	61.55	38.82	34.08	37%	45%	71.09	43.08	22.83	39%	68%
8	17.79	12.72	8.2	29%	54%	20.17	12.71	11.64	37%	42%
9	25.36	19.36	18.97	24%	25%	20.72	19.08	17.3	8%	16%

Note:
**Redu. CNN-ARIMA: denotes the reduced RMSE percentage difference between CNN and ARIMA with ARIMA RMSE as the denominator.*
**Redu. CNN-ARIMA: denotes the reduced RMSE percentage difference between RNN and ARIMA with ARIMA RMSE as the denominator.*

CNN performs similarly to LSTM when compared with the ARIMA model; it reduces the RMSE ranging from 8% to 63% and the MAE from 8% to 66%. In terms of the absolute mean error, the CNN model miss the prediction by 10 to 29 demand, which at most account for 7.5% of the observed demand among all series data.

Model Selection

Based on the significantly smaller input size and higher computational efficiency, this study selects the CNN model as the one for the demand forecasting task. Although LSTM produces slightly better prediction results, CNN, in the long run, would be much more applicable considering its ability to produce comparable prediction accuracy with observably reduced training expense.

The output of the prediction model will be inputted to the rebalancing optimization model to generate the final vehicle relocation plan.

Table 6- 6. Summary for MAE for proposed models

	Pickup					Drop-off				
Station index	ARI-MA	CNN	RNN	Redu. CNN-ARIMA	Redu. RNN-ARIMA	ARI-MA	CNN	RNN	Redu. CNN-ARIMA	Redu. RNN-ARIMA
1	38.06	18.09	15.02	52%	61%	21.95	14.66	16.07	33%	27%
2	18.59	11.86	10.16	36%	45%	23.58	13.19	11.99	44%	49%
3	33.31	27.44	23.05	18%	31%	25.38	21.23	17.10	16%	33%
4	29.51	21.08	17	29%	42%	38.25	26.25	26.41	31%	31%
5	47.19	16.14	11.93	66%	75%	18.51	14.62	12.49	21%	33%
6	26.84	15.24	13.2	43%	51%	25.90	20.82	19.00	20%	27%
7	49.22	29	25.35	41%	48%	57.50	31.76	18.74	45%	67%
8	14.31	10.38	6.9	27%	52%	15.63	10.40	8.93	33%	43%
9	20.29	14.71	15.46	27%	24%	16.77	15.37	14.54	8%	13%

Note:
**Redu. CNN-ARIMA: denotes the reduced RMSE percentage difference between CNN and ARIMA with ARIMA RMSE as the denominator.*
**Redu. CNN-ARIMA: denotes the reduced RMSE percentage difference between RNN and ARIMA with ARIMA RMSE as the denominator.*

6.3. Optimization Model for Rebalancing

This section presents the formulation of the rebalancing problem. This is the key step to determine the overall fleet relocation plan to serve the incoming demand better while minimizing the vehicle time spent on rebalancing tasks. Table 6-6 lists the notations for a set of key parameters.

Problem formulation

The objective function specified in Eq. (6-4) minimizes the total travel time for all relocated vehicles. This high-level objective considers real-world rebalancing scenario, in which the MOD vehicle drivers may be unwilling to perform the relocation task if they need to travel a long time to do so. Also, from a system point of view, it is unhealthy to assign more than necessary vehicles for rebalancing as it will cause increased vehicle mile traveled and generates more greenhouse gas emissions.

$$Min \sum_{i,j \in E} n_{ij} T_{ij} \tag{6-4}$$

However, this objective can be reformatted to reflect other rational concerns. For example, the total cost for rebabalcing work can be represented by changing the travel time, T_{ij}, to operational cost for each rebalancing task in the objective function.

The constraints include

1) For any stations, the assigned vehicles are no less than their needs.

$$\sum_j n_{ji} - \sum_j n_{ij} \geq n_i^{desired}(t+\Delta) - [V_i(t) + V_{ij}(t)] \tag{6-5}$$

The left-hand side is the sum of the total vehicles relocated to station *i* minus the sum of the total vehicles relocated from station *i*. The right-hand side is the predicted number of vehicles needed for station *i* in the look-ahead time, minus the sum of the vehicles serving station *i* and en-route to station *i* at the current time.

2) The predicted desired number of vehicles for all stations should be no more than the total fleet size.

$$\sum_{j} n_i^{desired}(t+\Delta) \le \sum_{i \in V}[V_i(t) + \sum_{j} V_{ij}(t)] \tag{6-6}$$

The left-hand side is the predicted sum of desired vehicles for all stations in the look-ahead time. And the right-hand side is the sum of all vehicles serving at stations and the ones taking rebalancing tasks.

Table 6- 7. Notations for rebalancing optimization formulation

Notations	
n_{ij}	Number of vehicles relocated from station *i* to station *j*
$\lambda_i(t)$	Number of requests at station *i* during time t
$q_i(t)$	Queued request at station *i* during time t
T_{ij}	Travel time from station *i* to station *j*
$V_i(t)$	Number of vehicles serving station *i* during time interval *t*
$n_i^{desired}(t)$	The required number of vehicles serving station *i* at time interval *t.* The output from the forecasting model.
$V_{ij}(t)$	Number of vehicles en route from station i to station *j* at time interval *t*
$G(V,E)$	A complete graph G with a set of stations in V and the edges between any pair of stations belong to E
Δ	Look-ahead time window

3) All relocated vehicles n_{ij} should be positive integers.

$$n_{ij} \geq 0, ij \in E \tag{6-7}$$

Given the formulation above, one can get a set of optimized n_{ij}, indicating how many vehicles need to be relocated from each station pair. Compared with the optimization model presented in the ride-sharing module, this model is much more computationally easy. Thus, the optimal gap for this model is set to 1% in all the simulation cases with rebalancing logic. The output of this MIP will be, for each vehicle, whether it is assigned to perform a rebalancing task and if so, to which station. This information will be used to update the vehicle information pool. To be more specific, vehicles with an on-going task will be asked to finish the current assignment and then perform a queued rebalancing task. If a vehicle is performing a relocating task, it will be ignored by the rebalancing module until it is no longer in the status of rebalancing. Note that, in this study, the vehicles are traveling without passengers onboard when performing rebalancing tasks. This assumption, however, can be relaxed in the future. For example, assigning vehicles to take passengers from one zone to another which concurrently, relocates the vehicles.

When the vehicle information pool is updated after the rebalancing work, this information combined with the predicted future demand stored in the request information poll will be inputted into the module -1 to begin the next iteration.

6.4. Summary

At a higher level, the rebalancing module is going to make better usage of the fleet size to prevent possible customer walk-aways due to long waiting times or insufficient supply of vehicles. This chapter presented the key components and their detailed methodology for the rebalancing module. To summarize, the rebalancing module consists of two core parts; one is the demand-forecasting model; the other one is the optimization model to generate an optimal vehicle relocation plan.

First, the demand forecasting model will be fed with past demand observations, and it generates the demand forecasting for each station. Acknowledging the rider demands are temporally correlated, three time-series models were developed and compared in three aspects that are crucial to ensure the success of the real-world application. The aspects are 1) prediction accuracy: measured by root mean squared error and mean absolute error; 2) input data size: how many past observations does the model need, and 3) Computational efficiency: whether it can produce results in a timely fashion to fit the online training purpose. Due to the proposed MOD, the system should be running in a non-closure service, and offline models cannot fit such purpose. Note that the ultimate goal of this chapter is to highlight the key concerns and general steps at the system planning stage. When it comes to the real-world application, we suggest that the operator perform more investigations on the seasonal or annual effect of the rider demands so as to better capture their inherent trends. Those effects can be helpful in improving the prediction power of the forecast model.

The output of the forecasting model is inputted into the optimization model, which was formulated as a mixed integer programming that minimizes the total travel time spent by conducting rebalancing task. This objective is set based on the reason that a vehicle will not serve any customers while conducting rebalancing task, thus minimizing this term is equivalent to maximizing the total usage of the available fleet. This model will generate a station based relocating plan that could best prepare the system for the incoming rider requests.

Chapter 7: Numerical Results

This chapter presents three numerical studies to assess the performance and applicability of the proposed system. All three numerical cases are carried out based on the Washington DC network. One case study employs the open taxicab data as the rider input, which intends to evaluate the system performance with short-distanced trips of medium demand level. The other two cases are fed with synthetic inputs of long-distance trips that mimic the anticipated popular use case for such a multi-modal transit model. Note that numerical case -1 tests the feasibility of the first two module's performance under varying demand levels, from medium to high. Its objective is to identify the changes in performance measures under different demand levels and evaluate the sensitivity of the proposed system through these changes. Based on the findings of case 1, numerical case 2 examines the usefulness of the rebalancing module under the simulated long-commuter trip patterns. Recognizing the synthetic data may not be able to fully present the real-world situation, numerical case 3 takes the DC taxicab data as the input and test the entire system's ability to handle real-world demand. For each case study, a set of performance measures are collected and summarized. The measures cover aspects of service efficiency, customer satisfaction, fleet utilization, and environmental benefits. Through extensive sensitivity analyses, some key trade-offs and complicated interaction among the three modules are further examined. To be more specific, these numerical tests are intended to address the following critical issues:

- Whether the proposed system can provide sufficiently good service to the riders, especially to those with fixed commuting routes that are highly overlapped with metro lines?
- How to select the proper measures of effectiveness in terms of both the service quality and environmental benefits?
- How do the key control parameters affect the system's performance and how does the influence reflect on the pre-set performance MOEs?
- What is the minimum fleet size that the proposed system requires to ensure adequate service performance?
- What is the proper adjustment to the method that can be made to enhance the service quality for the riders? How do these adjustments affect system performance?
- Whether the fleet operation would be more effective when the rebalancing module works collaboratively with the other two modules? If so, under what circumstance would it be most efficient?
- What types of demand pattern would enable the proposed system to deliver the highest emission savings?

Case one intends the answer the first five critical issues by simulating commuter demand of varying magnitude. To understand the fundamental interaction between rider demand and MOD supplies, this case study excludes additional fleet management strategies and only examines the fundamental ride-sharing methodology. Based on the findings from case 1, cases 2 and 3 further test the overall system performance with all three modules, highlighting additional performance enhancement brought about by the rebalancing module. The difference is that they are testing different demand patterns. Similar to case-1, case 2 simulates the long-commuter traffic while case-3 employs the real-world taxicab data. Case-3 stands for proof that the proposed system is capable of serving real-world demand and also examines the system's performance on short-distanced trip demand.

This chapter presents extensive simulation results to assess these criteria and is structured as follows. Section 7.1 introduces the simulation setting for numerical case-1, including the test network, simulation scheme and so on. Section 7.2 examines the performance of the proposed system and its potential benefits by conducting sensitivity analyses for a set of selected MOEs under high demand for long commuting trips. Section 7.3 and 7.4 summarize the simulation setting and results for case-2. The same set of performance measures used in case 1 are reported here. Section 7.5 introduces the simulation setting of the real-world demand, and Section 7.6 summarizes the system performance under medium level short distance trips and the comparison of system efficiency with and without the rebalancing module. Finally, Section 7.7 summarizes findings from both numerical tests and justifies the anticipated benefits of this research.

7.1. Numerical Case-1: Simulation Setting

This section presents the illustrative case under a set of hypothetical customer requests and MOD vehicle supplies at nine stations picked from Washington DC metro network. The main task of this numerical case study is to explore how the system performance will change under varying levels of customer demand and MOD vehicle fleet size. Because such an integrated system is still hypothetical, making sure its ability to handle a large amount of commuting traffic is the key to ensure its success in the future application. To fully understand the complicated relationship between its supply and demand, this case only assesses the performance of the map-partitioning and ridesharing modules. The effectiveness of rebalancing module will be accessed in the following two case studies using both synthetic long-distanced request data and real-world taxi data which consist of mainly short-distance trips. As shown in Figure 7-1 below, all selected stations are pined blue. Their latitude and longitude information was obtained from Google distance matrix API. The mass transit data were queried from WMATA API, which is sponsored by the DC metro operator and is used to provide publicly accessible metro information for developers. Real-time metro travel time and schedules of all selected stations were obtained directly from it.

Due to the absence of the real-world request data, this study generates the synthetic demand data across the sub-map that covers the Washington DC metro lines and examines the impact of total request number of varying sizes. Similar to the requests, the MOD vehicles' initial positions are also randomly generated across the map with the varying total fleet size. The distance between locations in the road network is obtained from Google distance matrix API.

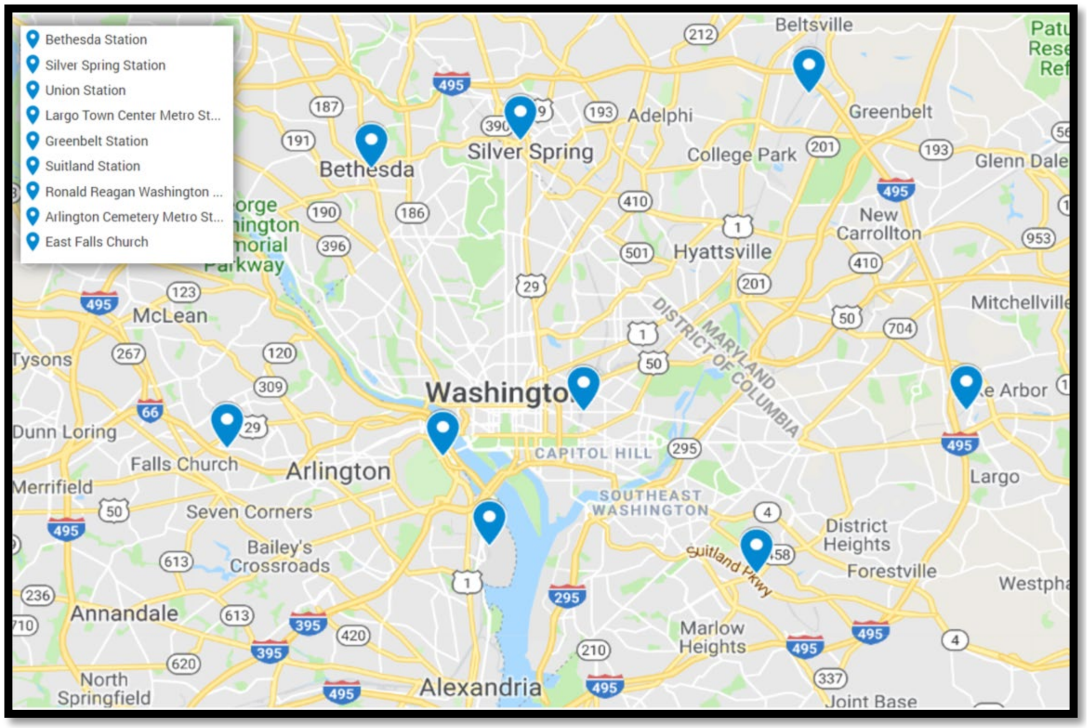


Figure 7- 1. Simulated metro station and road network for case-1

A set of key simulation settings are summarized as follows:

- Total MOD vehicle fleet size: 90, 180, 270, 360;
- Total Number of Requests (in one hr.): ranging from 5,000 to 12,000 with every 1,000 increments.
- Length of the iteration time window: 1 min, 3 min, and 5 min;
- Simulation Time and duration: Regular work-day from 8-9 am (1 hr simulation time of the morning peak)
- MOD vehicle maximum capacity: 3 passengers excluding the driver.

The simulation program is coded in *Python 3.4,* and *Gurobi* optimizer is used as the solver for the optimization model. In this study, the simulation scenarios are tested according to different setting combinations. All of the above-mentioned total requests and MOD fleet size are serving the selected nine stations. One-hour morning peak duration was applied to every testing scenario. To avoid other endogenous effects, the metro schedules were using the real-data from a non-holiday weekday from 8 am to 9 am. The MOD vehicles are assumed to be homogeneous regular passenger cars with the maximum capacity of passengers equal to three, excluding the driver. The real-time simulations were then conducted based on three types of updating intervals, including 1 min, 3 min, and 5 min. Besides the constraints presented in the ride-sharing module, this section further examines the system performance with and without maximum waiting time constraint, an additional constraint to the MIP model. This constraint is used to present the trade-off relationship between customer service quality and system efficiency. Note that all the results are obtained from real-time simulations.

7.2. Numerical Case-1: Simulation Results

This section selects a set of measures of effectiveness highlighting the proposed systems' efficiency in serving customers and its potential environmental benefits. The following presentations of the MOEs are based on the aggregate results of all the applicable sub-trips.

Number of Severed Requests

Figure 7-2 shows the comparison results for the number of served requests with respect to the total number of requests, MOD vehicle fleet size, and updating interval length. Given each fleet size and interval length combination, the number of served requests (y-axis) is plotted against the total number of generated requests (x-axis). One can observe that the number of served requests is increasing with the growing total demand size. Under a given demand, the number of served requests also increases with the rising total vehicle fleet size.

Not surprisingly, the largest fleet size with shortest interval length (fleet size 360 with interval length 1 min) outperforms the rest of the testing scenarios under all demand levels. This might be because, under such high-demand simulations, the 1 min interval is long enough for generating sufficient request for matching requests. Further, the shorter interval allows the available vehicle information updates more frequently, which are more likely to be matched with riders compared with the larger interval lengths. However, shorter interval does not always outperform the longer intervals. For example, under the smallest fleet size (fleet size 90), the 3 min interval generates better results than 1 min, and 5 min interval, meaning that more matching opportunities can be founding using 3 min interval length when the total fleet size equals 90.

In conclusion, there is a tradeoff between the fleet size and the updating interval length. On the one hand, the longer the interval, the more opportunities exist for the request-to-request ride-sharing match. On the other hand, the shorter the interval, the more likely it is to achieve the vehicle-to-request match. When fleet size and demand size are both large, the shorter interval outperforms the longer ones. However, for the rest of the cases, one has to balance between those two matches (request-to-request, vehicle-to-request) for the selection of interval length.

One more observation is the need for rebalancing, as one can see from the plot, if the fleet size increase from 90 to 180, the served request line is on average, increased around 1500 to 3000 served requests across all demand levels. Such an increment is not seen by increasing the fleet size from 270 to 360 where the results are mostly similar to each other regardless of the interval length. This is indicating that this increased fleet size is not properly utilized, as there are still plenty of unserved requests. Relocating the under-utilized vehicles to areas with more unserved demand can help with this situation.

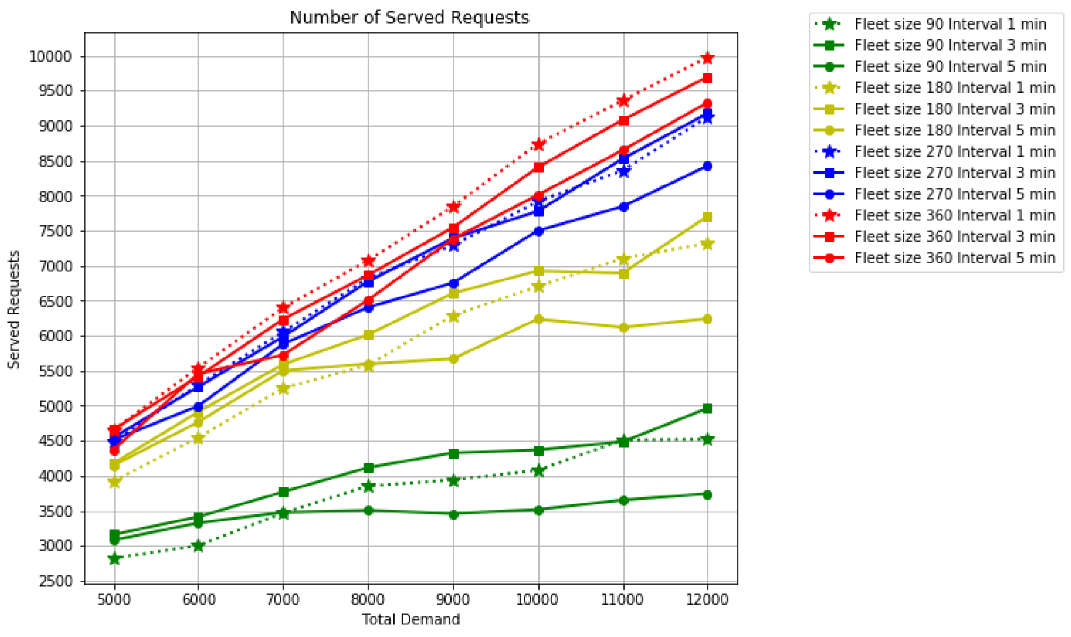


Figure 7- 2. Comparison of the results based on the number of served requests

Total Emission Savings

One of the most promising benefits of the proposed system is the emission savings for riders shifted from private car usage to this multi-modal shared mobility.

The following calculation method is adapted from the US Environmental protection agency:

$$\begin{aligned}&\text{Total Emission Saving=}\\&8.89\times10^{-3}\text{ metric tons/ gallon gasoline}\times\Delta_{Total\text{ vehicle mile savings}}\times\\&(1/22)\text{ miles/gallon}\times1\ CO_2, CH_4\text{ and }N_2O/0.989\ CO_2\end{aligned} \tag{7-1}$$

The emission savings is measured in metric tons CO_2E. The amount of carbon dioxide emitted per gallon of motor gasoline burned is 8.89×10^{-3} metric tons. In 2015, the ratio of the carbon dioxide emissions to the total greenhouse gas emissions (including carbon dioxide, methane, and nitrous oxide, all expressed as carbon dioxide equivalents) for passenger vehicles was 0.989 (EPA 2017). The ratio 1/22 miles per gallon is the average car miles traveled per gallon gas. The $\Delta_{Total\text{ vehicle mile savings}}$ is the total miles traveled saving defined as the difference between total vehicle miles if traveled directly and the total miles traveled by all MOD vehicles.

The results are presented in Figure 7-3. One can clearly observe that emission saving is strictly positive under all testing scenarios. As expected, emission saving increases with the increase in the total demand and fleet size. These results prove that the proposed system can significantly reduce vehicular emissions to benefit society environmentally.

Average Number of Served Requests per Trip

The third key performance measure is the average number of served requests per trip, which is calculated as the total number of served requests divided by the total number of trips. Under each testing scenario, this measure is taken as the average of both the 1st and 3rd trips. Table 7-1 summarizes the average number of served requests per trip under the same set of testing scenarios.

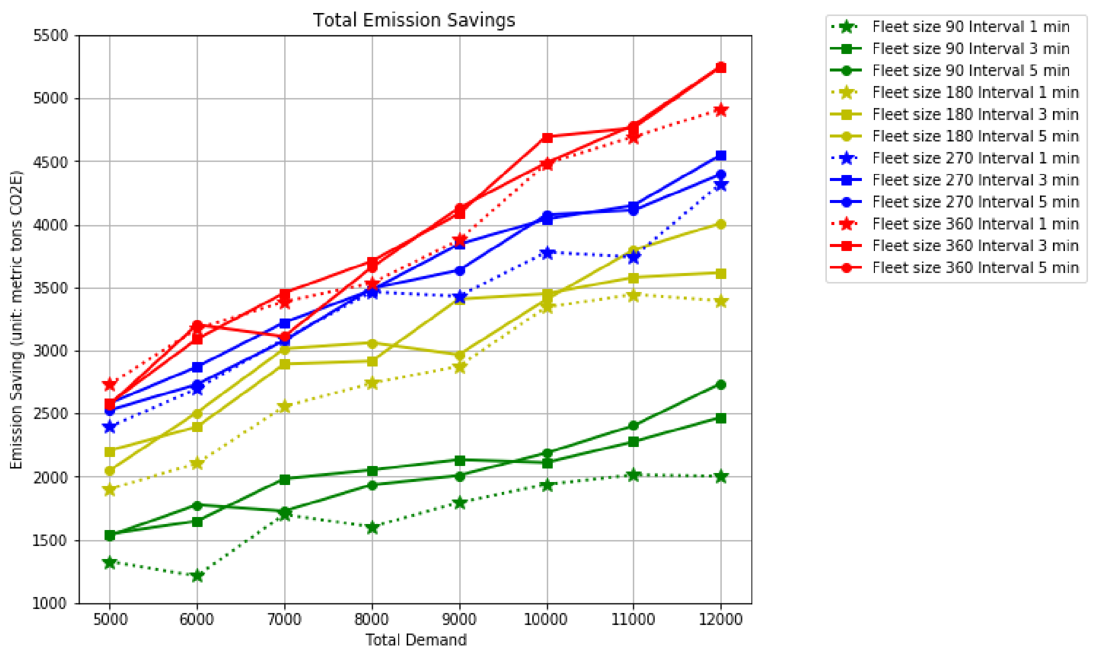


Figure 7- 3. Comparison of the results for total emission savings

From the results, one can see that under the same fleet size, the average number of served requests per trip increases with the increase of demand level, which means more ride-sharing are made. On the other hand, under the same demand, this term also increases with the increase of interval length. This finding is as expected, the longer the interval length, the more opportunities to find request-to-request matches. Theoretically, the maximum average number of served requests per trip should be no more than the maximum vehicle capacity, which is three.

Table 7- 1. Summary of the average no. of served requests per trip

	Average no. of served requests per trip											
Demand	fleet 90			fleet 180			fleet 270			fleet 360		
level	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***
5000	1.24	1.53	1.68	1.23	1.46	1.58	1.22	1.46	1.61	1.25	1.47	1.64
6000	1.32	1.59	1.74	1.26	1.52	1.63	1.26	1.5	1.61	1.28	1.47	1.65
7000	1.34	1.66	1.82	1.27	1.52	1.64	1.29	1.48	1.64	1.28	1.51	1.59
8000	1.35	1.69	1.88	1.33	1.59	1.68	1.3	1.51	1.66	1.31	1.53	1.65
9000	1.42	1.71	1.89	1.36	1.57	1.74	1.33	1.55	1.68	1.32	1.52	1.61
10000	1.45	1.79	1.98	1.37	1.62	1.78	1.34	1.56	1.72	1.34	1.55	1.62
11000	1.49	1.89	2.01	1.41	1.67	1.77	1.38	1.59	1.71	1.35	1.58	1.65
12000	1.5	1.8	1.99	1.44	1.67	1.81	1.37	1.57	1.69	1.35	1.56	1.66

Average Request Waiting Time-for MOD vehicles

The average request waiting time is a direct indicator to evaluate the customer service quality. It is defined as the difference between customer request start time and the customer pick-up time. This measure is also taking an average of 1st and 3rd trip aggregately under each testing scenario. Some existing literature provides constraints on the maximum customer waiting time to provide satisfactory service quality as real-world users might walk-away if their requests are not served within their tolerance limit. This study examines the scenarios with and without this constraint and presents the trade-off relations among the service quality of the served requests and the system efficiency.

Table 7-2 shows the average customer waiting time for MOD vehicles under different scenarios. Note that all served request satisfied their requested arrival time window no matter how long they wait for the MOD vehicles to pick them up. From this table, the mean waiting time is around 15 minutes for all simulations. Acknowledging such a long waiting time may induce customer dissatisfaction, and they might walk away, this study further presents the results with the waiting time constraints.

Table 7- 2. Summary of the average customer waiting time for MOD vehicles

Average customer waiting time-MOD vehicles (minutes: seconds)												
Demand level	fleet 90			fleet 180			fleet 270			fleet 360		
	1min	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***
5000	11:52	14:12	14:45	12:55	14:46	15:54	13:42	15:25	16:12	13:51	15:29	16:10
6000	12:10	14:28	15:39	12:36	15:06	16:07	13:21	15:25	16:41	13:53	15:58	16:31
7000	12:11	14:41	15:40	13:00	15:31	16:18	13:51	15:58	16:29	14:03	15:54	16:01
8000	12:24	14:32	15:50	13:07	15:34	15:56	13:38	15:41	16:30	14:24	15:56	16:27
9000	12:47	15:00	16:07	13:12	15:29	16:26	13:49	15:52	16:32	14:15	16:06	16:12
10000	12:50	14:59	15:43	13:18	15:27	16:29	13:54	16:01	16:21	14:19	16:11	16:15
11000	13:14	15:16	16:27	13:34	15:34	16:15	14:08	15:57	16:35	14:25	16:11	16:30
12000	12:51	14:58	15:58	13:45	15:35	16:23	14:07	15:41	16:23	14:14	15:46	16:08

Average Request Waiting Time-for metro

Table 7-3 lists the average customer waiting time for metro for all test scenarios. Under the same demand level and fleet size, the metro waiting time is the smallest with a 5-minute interval length. On the other hand, under the same demand level and the interval length, the scenarios with large fleet size generate trips with shorter metro waiting time.

Other MOEs

Table 7-4 summarizes the average request travel time using MOD vehicles. One may notice that the travel time remains similar across all the test scenarios. Note that this is different from the average MOD vehicle travel time per trip since a trip can consist of multiple requests.

Table 7- 3**.** Summary of the average customer waiting time for metro

Average customer waiting time-metro (minutes: seconds)												
Demand level	fleet 90			fleet 180			fleet 270			fleet 360		
	1min	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***
5000	14:12	11:06	9:50	11:57	9:18	8:09	10:27	8:05	7:01	9:39	7:49	7:08
6000	14:06	11:02	9:16	12:25	9:19	7:32	10:46	8:09	6:53	9:49	8:17	8:24
7000	14:14	10:28	8:40	12:25	8:50	7:40	10:35	7:48	6:35	9:45	7:27	7:51
8000	13:47	10:17	8:12	12:19	8:45	7:33	10:53	7:50	6:16	9:34	7:12	6:31
9000	13:35	10:06	8:16	12:02	9:00	7:11	10:54	7:39	6:36	9:48	7:14	7:06
10000	13:50	9:35	8:10	12:09	8:50	6:59	10:54	7:31	6:22	9:52	7:07	8:06
11000	13:04	9:27	7:48	11:55	8:42	7:17	10:45	7:18	6:15	9:45	6:57	6:38
12000	13:21	9:47	7:45	11:46	8:08	7:05	11:08	8:11	7:05	10:28	7:45	6:57

Table 7- 4. Summary of the average request travel time in MOD vehicles

Average request travel time in MOD vehicles (minutes: seconds)												
Demand level	fleet 90			fleet 180			fleet 270			fleet 360		
	1min	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***
5000	7:10	7:52	8:06	7:28	8:10	8:11	7:45	8:19	8:23	7:58	8:20	8:32
6000	7:10	7:54	7:58	7:27	8:05	8:09	7:41	8:12	8:12	7:59	8:17	8:24
7000	7:19	7:46	7:58	7:40	7:59	8:05	7:47	8:10	8:12	7:56	8:21	8:21
8000	7:19	7:41	7:45	7:34	7:54	7:58	7:46	8:10	8:13	7:55	8:18	8:19
9000	7:32	7:40	7:44	7:35	7:59	7:53	7:47	8:06	8:08	7:47	8:16	8:07
10000	7:26	7:41	7:41	7:36	7:55	7:52	7:48	8:05	8:04	7:52	8:09	8:06
11000	7:29	7:41	7:39	7:36	7:58	7:49	7:49	7:59	7:55	7:52	8:09	8:09
12000	7:21	7:39	7:22	7:37	7:44	7:49	7:47	7:59	7:53	7:52	8:09	8:05

Similar results are also found in average request in-metro travel time, shown in Table 7-5. The average in-metro travel time is around 1hr across all simulation scenarios. This result matched the station-to-station travel time obtained from google maps.

Table 7- 5. Summary of the average request travel time in metro

Average request metro travel time (minutes: seconds)												
Demand level	fleet 90			fleet 180			fleet 270			fleet 360		
	1min	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***
5000	58:03	57:57	56:27	58:04	57:50	57:55	57:53	58:08	57:05	57:40	57:31	57:34
6000	58:21	57:52	57:24	57:56	57:40	57:36	57:27	57:28	57:18	57:50	57:35	57:44
7000	58:14	57:35	56:14	57:57	57:45	57:54	57:18	57:34	57:48	57:42	57:26	57:40
8000	57:49	56:42	55:56	57:46	58:03	57:03	57:29	57:39	57:40	57:48	57:25	57:35
9000	58:23	57:56	55:45	57:40	58:05	56:43	57:45	57:31	57:30	57:32	57:39	57:27
10000	59:05	57:25	55:08	58:04	58:25	57:35	57:43	57:21	57:25	57:35	57:50	57:34
11000	58:34	57:47	56:00	58:25	57:36	55:57	58:00	57:36	57:20	57:44	57:51	57:31
12000	58:15	57:40	54:36	58:19	57:42	56:03	57:38	57:34	57:21	57:38	57:42	57:13

Table 7-6 summarizes the average total request travel time.

This measurement is approximately around 1 hr and 30 minutes for all test scenarios, which is slightly higher than the multi-modal travel time without ride-sharing option obtained from google map.

Table 7- 6**.** Summary of the average total travel time per request

Average total request travel time (hour: minutes)												
Demand level	fleet 90			fleet 180			fleet 270			fleet 360		
	1min	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***	***1min***	***3min***	***5min***
5000	1:34	1:36	1:35	1:33	1:35	1:36	1:32	1:35	1:35	1:32	1:34	1:35
6000	1:35	1:36	1:36	1:33	1:35	1:35	1:32	1:34	1:35	1:33	1:35	1:35
7000	1:35	1:36	1:35	1:34	1:35	1:36	1:33	1:35	1:35	1:33	1:34	1:36
8000	1:35	1:35	1:34	1:34	1:36	1:35	1:33	1:35	1:35	1:33	1:34	1:35
9000	1:36	1:36	1:34	1:34	1:36	1:35	1:34	1:35	1:35	1:33	1:35	1:35
10000	1:37	1:36	1:33	1:34	1:36	1:35	1:34	1:35	1:35	1:33	1:35	1:35
11000	1:36	1:36	1:35	1:35	1:37	1:36	1:35	1:35	1:35	1:34	1:35	1:35
12000	1:36	1:36	1:33	1:36	1:35	1:34	1:35	1:35	1:35	1:34	1:35	1:35

Besides the table of frequencies for the performance measures presented above, the optimization goodness obtained by *Gurobi Optimizer* is also checked. The optimal gap is generally below 10% for each iteration.

Impact of adding waiting time constraint (WTC)

Uber (Myhrvold, 2015) supports that a customer walks away if they wait at least 6 minutes. To ensure customer satisfaction, the maximum waiting time constraint equal to 6 min was applied to each customers' 1st and 3rd sub-trips. Even if the system can serve the request without violating its arrival time window constraint, the request is declined once the waiting time exceeds $t_{\max} = 6\,\text{min}$. This constraint, however, is not applied to the in-metro sub-trip but only for ones involving MOD vehicles. This section is using a fixed fleet size of 270 with varying demand level to evaluate system performance with and without waiting time constraint.

1) Average Request Waiting time

Table 7-7 summarizes the average request waiting time for MOD vehicles with and without WTC. The results clearly indicate the constraint is successfully imposed.

Table 7- 7**.** Comparison of the average request waiting time w./w.o. WTC

Average request waiting time-MOD vehicles (minutes: seconds)						
Demand level	No WTC			With WTC=6 min		
	1min	*3min*	*5min*	*1min*	*3min*	*5min*
5000	13:42	15:25	16:12	4:42	4:47	5:08
6000	13:21	15:25	16:41	4:38	4:56	4:51
7000	13:51	15:58	16:29	4:32	4:55	5:05
8000	13:38	15:41	16:30	4:33	5:00	5:02
9000	13:49	15:52	16:32	4:33	4:56	5:04
10000	13:54	16:01	16:21	4:33	4:54	5:03
11000	14:08	15:57	16:35	4:26	4:55	5:08
12000	14:07	15:41	16:23	4:32	4:59	5:12

2) Number of Served Requests

Figure 7-4 shows the results of the comparison of the number of served requests with and without waiting time constraint. Under the fixed fleet size of 270, the existence of WTC significantly reduces the total number of served requests. This reduction happens under all three interval length settings.

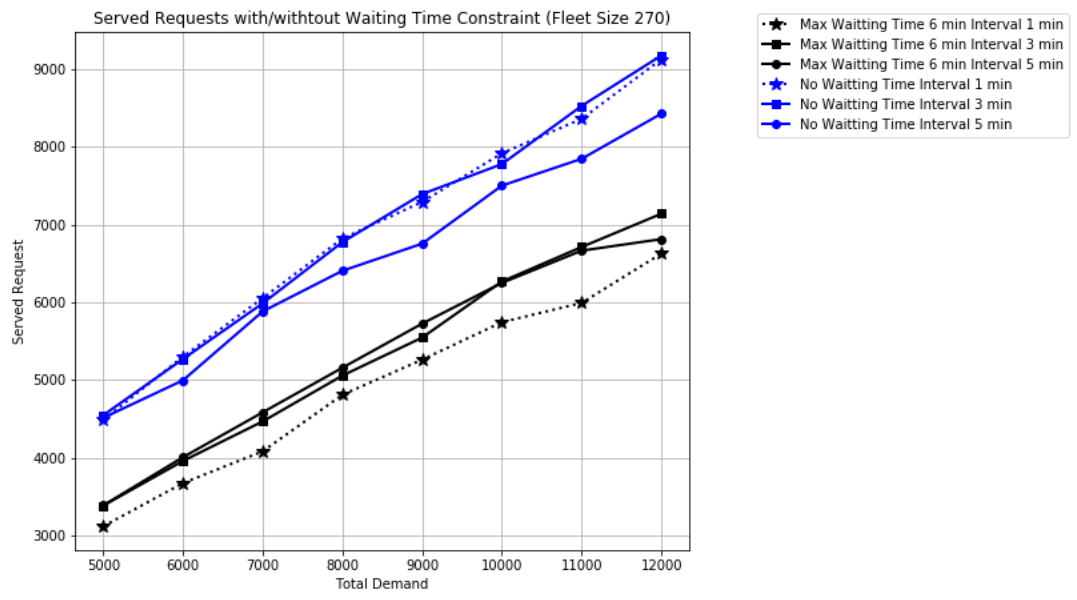


Figure 7- 4. Comparison of the number of served requests w./w.o. WTC

3) Total Emission Savings

Similar to the case for the total number of served requests, the emission savings also exhibit an obvious reduction (Figure 7-5). This decrement is consistent with all simulation interval lengths.

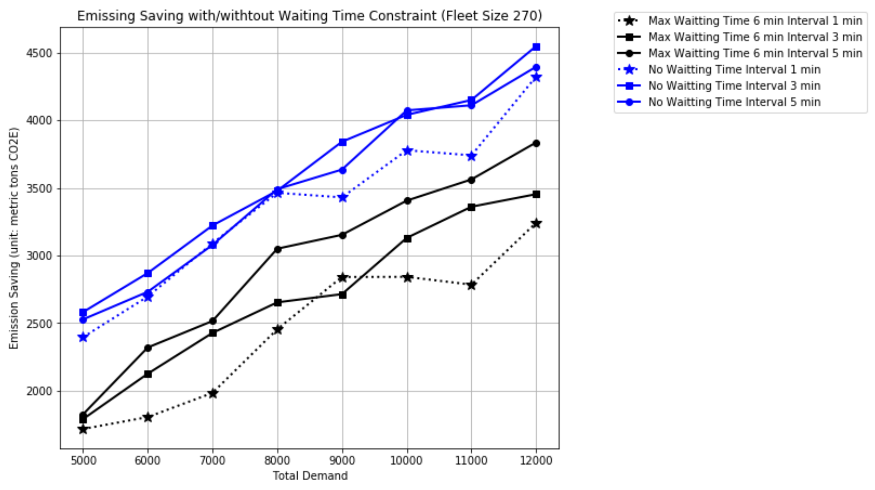


Figure 7- 5**.** Comparison of the total emission savings w./w.o. WTC

4) Average Number of Served Requests per Trip

Table 7-8 shows the comparison of the average number of served requests per trip with and without waiting time constraints. Imposing this constraint limits many opportunities for potential ride-sharing. However, this negative impact diminished under high demand cases. For example, the average number of served requests per trip with WTC is 1.7 when total demand equals 12,000, which is like the 1.72 average number of served requests per trip without WTC.

Table 7- 8. Comparison of the average no. of served requests per trip w/w.o. WTC

Average number of served requests per trip						
Demand level	No WTC			With WTC=6 min		
	1min	*3min*	*5min*	*1min*	*3min*	*5min*
5000	1.22	1.46	1.61	1.12	1.34	1.49
6000	1.26	1.5	1.61	1.14	1.36	1.53
7000	1.29	1.48	1.64	1.16	1.39	1.57
8000	1.3	1.51	1.66	1.18	1.44	1.62
9000	1.33	1.55	1.68	1.19	1.45	1.67
10000	1.34	1.56	1.72	1.21	1.49	1.7
11000	1.38	1.59	1.71	1.21	1.53	1.73
12000	1.37	1.57	1.69	1.23	1.55	1.74

5) Other MOEs

This study further compared the WTC's effect over the following MOEs: 1) average Customer waiting time for metros; 2) Average request travel time in MOD vehicles; 3) average request travel time in metro, and; 4) average total request travel time. Their results are presented in Tables 7-9 and 7-10.

Table 7- 9**.** Comparison of the average request metro waiting time w/ w.o. WTC

Average request waiting time-metro (minutes: seconds)						
Demand level	No WTC			With WTC=6 min		
	1min	*3min*	*5min*	*1min*	*3min*	*5min*
5000	10:27	8:05	7:01	7:48	6:53	7:01
6000	10:46	8:09	6:53	7:54	7:18	7:41
7000	10:35	7:48	6:35	8:07	7:40	7:36
8000	10:53	7:50	6:16	8:28	7:49	8:02
9000	10:54	7:39	6:36	8:48	8:11	8:17
10000	10:54	7:31	6:22	8:59	8:25	8:25
11000	10:45	7:18	6:15	8:34	8:20	8:33
12000	11:08	8:11	7:05	9:02	8:37	8:32

Table 7- 10. Comparison of the average request travel time w/ w.o. WTC

(a) Average request travel time in MOD vehicles

Average request travel time-MOD (minutes: seconds)						
Demand level	No WTC			With WTC=6 min		
	1min	*3min*	*5min*	*1min*	*3min*	*5min*
5000	7:45	8:19	8:23	7:22	7:47	7:53
6000	7:41	8:12	8:12	7:20	7:41	7:58
7000	7:47	8:10	8:12	7:23	7:46	8:03
8000	7:46	8:10	8:13	7:25	7:53	7:59
9000	7:47	8:06	8:08	7:25	7:43	8:01
10000	7:48	8:05	8:04	7:26	7:45	7:59
11000	7:49	7:59	7:55	7:16	7:45	7:59
12000	7:47	7:59	7:53	7:24	7:43	8:04

(b) Average request travel time in metro

Average request travel time-metro (minutes: seconds)						
Demand level	No WTC			With WTC=6 min		
	1min	*3min*	*5min*	*1min*	*3min*	*5min*
5000	57:53.0	58:08.0	57:05.0	57:54.0	56:57.0	56:53.0
6000	57:27.0	57:28.0	57:18.0	57:12.0	56:52.0	56:33.0
7000	57:18.0	57:34.0	57:48.0	57:22.0	57:12.0	57:15.0
8000	57:29.0	57:39.0	57:40.0	56:50.0	56:50.0	56:52.0
9000	57:45.0	57:31.0	57:30.0	56:37.0	57:16.0	57:19.0
10000	57:43.0	57:21.0	57:25.0	57:22.0	57:06.0	56:33.0
11000	58:00.0	57:36.0	57:20.0	56:56.0	57:00.0	56:30.0
12000	57:38.0	57:34.0	57:21.0	56:57.0	56:53.0	56:15.0

(c) Average request total travel time

Average total request travel time- MOD + metro (hours: minutes)						
Demand level	No WTC			With WTC=6 min		
	1min	*3min*	*5min*	*1min*	*3min*	*5min*
5000	1:32	1:35	1:35	1:24	1:24	1:25
6000	1:32	1:34	1:35	1:24	1:24	1:26
7000	1:33	1:35	1:35	1:24	1:25	1:26
8000	1:33	1:35	1:35	1:24	1:25	1:26
9000	1:34	1:35	1:35	1:26	1:26	1:27
10000	1:34	1:35	1:35	1:25	1:26	1:26
11000	1:35	1:35	1:35	1:24	1:26	1:26
12000	1:35	1:35	1:35	1:25	1:26	1:26

Similar to the results shown above, they are stable and stay unchanged regardless of the existence of the waiting time constraints. These match the expectation that this strong constraint will mostly affect the MOD vehicle average customer waiting time, average served requests per trip, emission savings and the number of served requests.

The imposition of WTC can greatly affect the system's key performance measures. This constraint can effectively reduce the customer waiting time, which is commonly used to define customer satisfaction. However, the opportunities for ride-sharing are negatively influenced, and fewer requests can be combined and served in one vehicle. This finding inspires the decision maker to find a balanced solution between solution efficiency and service quality.

Closure

This section presented extensive simulation results from numerical case-1 to validate the theoretical work of the proposed system. In this case study, we generated synthetic demand data to test the system performance under varying levels of demands and MOD supplies. The generated requests are long distance requests that are at least from the suburban areas to the center of DC to evaluate the system's ability to serve as a private car alternative for committing purpose. Also, to match the magnitude of daily commuting trips, high levels of random requests are generated. This section collected several performance measures that characterize the system, including the number of served requests, total emission saving, average served requests per trip, average customer waiting time for the MOD vehicle and so on. For each demand level, four different fleet sizes with three different lengths of simulating intervals are tested against a set of MOEs. In general, the system efficiency, in terms of the number of served requests, emission savings, the average number of served requests per trip, is increased with the larger fleet size if demand is fixed. Further, the updating interval length will also significantly impact the system performance, as the shorter duration usually performs better under high demand level given a fixed MOD fleet size. To summarize, this case study justifies the ability of the proposed theoretical work to serve high-level commuter requests if the fleet size and updating interval length are properly chosen.

Two trade-offs were identified. One is the vehicle-to-request match opportunity that is going to increase with shorter updating intervals. Another one is the request-to-request ridesharing opportunities that have more chances for matching when longer intervals are used. Besides the system efficiency concern, this case study also examined the trade-offs between service quality and system efficiency, represented by the maximum waiting time enforcement. However, regardless of how system control parameters are set, the proposed model (map-partitioning + ridesharing) can significantly reduce greenhouse gas emission caused by private car commuting, thus justifying the legitimacy of this theoretical work.

The next case study will test the reliability of the rebalancing module on top of the first two modules, as tested in case study-1. The same set of the MOEs is reported and compared under cases with varying fleet sizes but for a fixed demand level.

7.3. Numerical Case-2: Simulation Setting

Based on the key findings from the previous case, numerical case 2 is mainly designed to explore the effectiveness and reliability of the rebalancing module on top of the first two modules. Case -1 already shows that the first two modules can collaboratively work well to serve simulated long-commuter requests of high demand. The ultimate purpose of the inclusion of the rebalancing module is to further enhance their performance through better utilization of the available supply. This relocation, if designed properly, can be extremely helpful to facilitate the service for commuter traffic, as most requests happen in the morning or afternoon peak hours. Relocation of the vehicles according to the time-varying demand pattern would be beneficial to serve customers in queue and drivers seeking the best opportunity to make money. The main task of this case study is to examine how much benefit can the rebalancing bring to the system and how does such benefit reflect on the selected MOEs. Note that case study 3 also examines the same aspect but the major difference is that case 2 is tested under long-commuter requests, but case 3 applied to the short-distance taxi trip data.

This case study partially adopts the same simulation settings as in case 1, including the selection of nine stations (as shown in Figure 7-1), requesting travel time information from google map API and metro schedules from WMATA API. Note that to simulate the imbalanced long-distance commuter demands, this case study adopts the demand pattern of taxi cab data by pairing the two different set of stations. The stations used in case 3 (shown in Figure 6-3) and stations used in case 2(shown in Figure 7-1) are paired based on their geometrical closeness. For each station in case 2, the demand forecast model is the same as its paired stations in case 3.

The reason to do this is the author believes that adopting the real-pattern is better than generating random OD pairs without any basis. The second reason is, by pairing the stations, the rebalancing module can still utilize the results from the proposed demand forecast model, which is the core part constructing the rebalancing module. A randomly simulated demand can hardly be predictable, as the random data fails to exhibit the daily repeatable pattern. However, if we simulate a repeatable pattern with underlying distribution, such data can often fail to present the real-world uncertainty, which in a sense, contradicts to the biggest challenges faced by prediction models and the rebalancing module.

After obtaining the hourly pick-up and drop-off volume at each station, this case randomly generates the coordinates of requests around each station. The total number of station-based requests matches the one from the taxi cab data. Like the case-1, the initial position of the MOD vehicles is also generated randomly across the map. To overcome the potential bias due to such randomness, three random seeds are used, and the average of them is reported as the results.

Besides that, the following are the set of key control parameters:

- Total MOD vehicle fleet size: 270, 360, and 450;
- Length of the iteration time window: 5 min, 10 min, and 15 min;
- Simulation Time and duration: 6:00 am to 6:00 pm for July 27th, 2017.
- MOD vehicle maximum capacity: 3 passengers excluding the driver.

The simulation program is coded in *Python 3.4,* and *Gurobi* optimizer is used as the solver for the optimization model. The optimal gap for the ride-sharing MIP is below 10% and below 1% for the rebalancing MIP. The maximum waiting time constraint is imposed to the 1st sub-trip of all requests, as it is reasonable to assume that the rider would stick with the planned trip if he doesn't walk away at the 1st part. Also, from case-1 results, one can see that having such constraint helps to ensure the quality of service, so case 2 and 3 both imposed it.

Because this case study applies the request demand with the same magnitude of taxicab data, it is in no way comparable to the simulated high demand levels in case 1. This case study adopts relatively longer updating intervals to ensure that enough request-to-request match opportunities. Similar updating interval lengths were also employed in case 3. The following section will report the results of the selected performance measures.

7.4. Numerical Case-2: Simulation Results

Note that to test the potential benefits of implementing the rebalancing module, this simulation assumes 100% compliance rate for MOD vehicles to take the relocation order. The following presentations of the MOEs are based on the aggregate results of all applicable sub-trips. Under the same demand level, this case highlights the difference between with and without rebalancing module.

Number of Severed Requests

Figure 7-6 shows the comparison results of the number of served requests between with and without rebalancing module testing cases. Given each fleet size and interval length combination, the number of served requests (y-axis) is plotted against the total available fleet size (x-axis). Similar to the observation in case-1, under a given demand, the number of served requests increases when the available fleet size increases.

In this case, we can clearly see that the shortest simulation interval length outperforms the rest. This might because the shorter intervals can have more opportunity for the vehicle-to-request match, thus result in more served requests compared to other scenarios. However, one can observe that there is not much difference between with and without rebalancing modules under 5 min interval length.

In terms of the performance enhancement coming from rebalancing module, one can clearly see that such efficiency improvement increases with the increase of interval length. Under the same interval length, the rebalancing benefit becomes more obvious with the larger fleet size. This observation makes sense because if a vehicle is assigned a rebalancing task, it is not able to serve any requests until it arrives at the destination station. Longer interval indicates more chances for vehicles to arrive at the destination station thus adding more available vehicles for service.

One thing to further notice is that the 15 min, which is the longest testing interval length, with the rebalancing module performs similarly good to the ones with 5 min interval. However, this interval length without the rebalancing module performs much worse in terms of the number of served requests. This is another trade-off that exists between the opportunities to match vehicles to requests and availability of vehicles for rebalancing. Since only vehicles without rebalancing work can be matched with requests, shorter interval length often provides more chances. However, the longer the interval, the more likely vehicles rebalanced during the last interval would arrive at the destination station. Thus they can be added to the available fleet to enhance the service opportunity.

Note that, because the main objective is to explore the potential benefits of the rebalancing module, the testing fleet sizes in this case study is adequate so that one could see the impact of this additional logic when it is fully functional. However, next case study 3 will examine the performance of scenarios when the fleet size is not adequate.

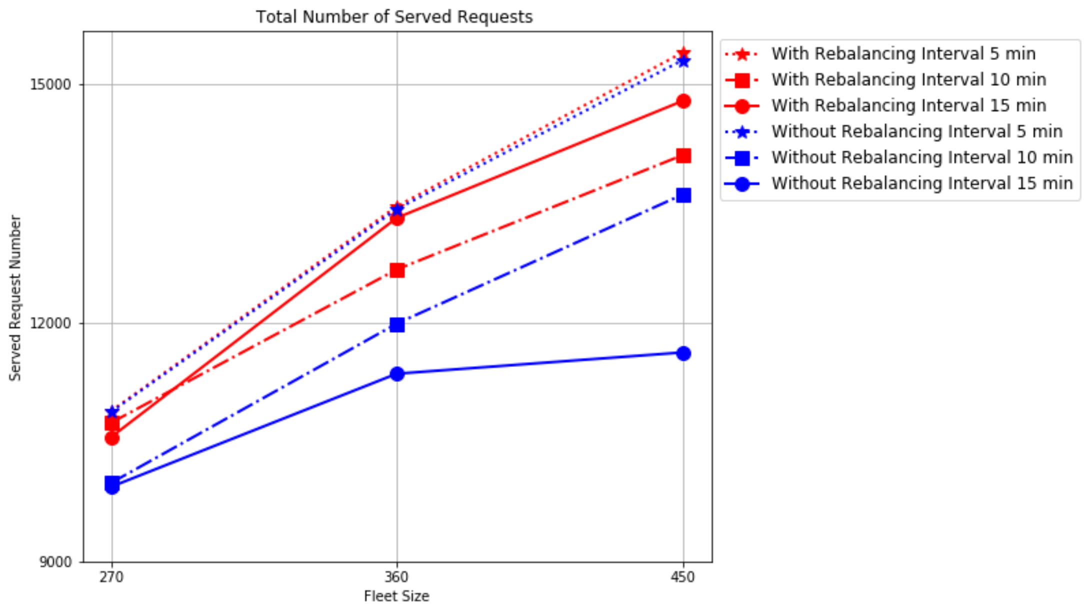


Figure 7- 6. Number of served requests w/w.o rebalancing module for case-2

In summary, the rebalancing module can improve the number of served requests, which is a core metric to measure system efficiency. Its improvement will become more obvious when the interval length and fleet size are of adequate size. Under a shorter updating interval, the effectiveness of the rebalancing module can become marginal when compared with the case without it. There is apparently, a trade-off between non-rebalancing vehicle-to-request match and the improvement coming from rebalancing vehicles. The former will become easier with shorter interval length while the longer interval length helps to have more available vehicles completing the rebalancing task.

Total Emission Saving

One of the most promising benefits coming from the proposed system is the reduction in greenhouse gas emissions, which is calculated using Eq. (7-1). The emission savings shown in Figure 7-7 is measured using metric tons CO2E. However, for the cases with the rebalancing module, the term $\Delta_{Total\ \text{vehicle mile savings}}$ total miles traveled saved, is defined as the difference between total vehicle miles for vehicles traveling directly minus the vehicle miles traveled in simulations. This is the sum of MOD vehicle traveled serving requests and conducting the rebalancing tasks.

From the graphical results, one can see that the 15 min updating interval with the rebalancing logic delivers the most emission saving. Similar to the finding from the number of requests served, the cases with rebalancing outperform the ones without it consistently. The longer the interval length is, the more emission savings is delivered by the rebalancing module.

Besides the emission saving, the parking cost saving should also be considered. The average daily parking cost in DC is \$25. When taking this rate into account, the parking cost saving for the commuters can range from 0.23 to 0.42 million dollars per day in total. That would also count as the benefit brought by this system.

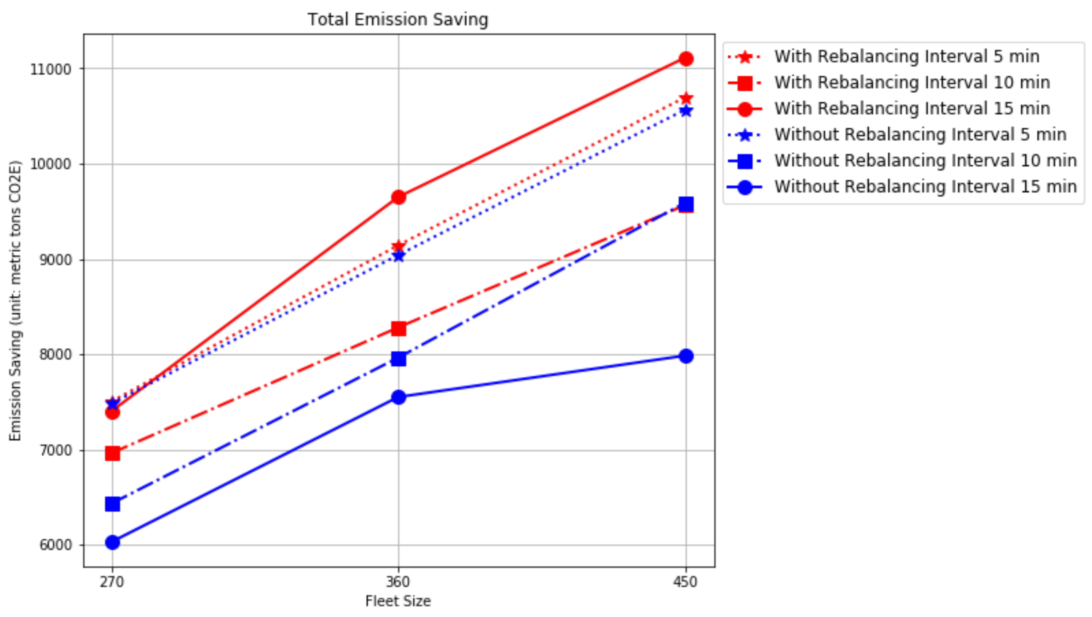


Figure 7- 7. Total emission saving w/w.o rebalancing module- case 2

Average Number of Served Requests per Trip

Table 7-11 summarizes the average number of served requests per trip as well as its standard deviation given different fleet size and interval length combinations. Similar to the findings from analyzing the number of served requests, the average number of served requests per trip is increasing with the increase in the interval length under the same fleet size. Also, when talking about the impact of the rebalancing module on the average number of served requests per trip, one can see that the ones with the rebalancing module outperform the ones without it. Also, similar to the previous two observations, the longer the updating interval length is, the more improvement is delivered by the rebalancing module.

Table 7- 11. Summary of the average no. of served requests per trip- case 2

Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 270	5 min	1.48	0.87	1.48	0.86
	10 min	1.9	1.02	1.88	1.02
	15 min	2.09	1.08	2.03	1.06
fleet 360	5 min	1.54	0.87	1.54	0.88
	10 min	1.94	0.94	1.91	0.99
	15 min	2.14	1	2.06	1
fleet 450	5 min	1.58	0.87	1.57	0.87
	10 min	1.94	0.95	1.93	0.95
	15 min	2.14	0.9	2.05	1

Utilization Rate

Besides the average number of served requests per trip, the utilization rate for MOD vehicles can be another metric to assess the fraction of time that the fleet is in productive use. In this study, it is defined as:

$$\text{Utilization Rate}= \frac{\text{Total vehicle travel time with passenger onboard}}{\text{simulation duration* total fleet size}} \tag{7-2}$$

where the total vehicle travel time with passengers onboard records the sum of the time of MOD vehicle with at least one passenger onboard. This term excludes the time of empty vehicle traveling, including when the vehicle is not assigned with any task, when the vehicle is conducting rebalancing task or when the vehicle is on the way to the next pickup location without passenger onboard. The results are presented in Table 7-12.

Table 7- 12. Summary of utilization rate for MOD vehicles - case 2

Fleet size	Interval length	With rebalancing	Without rebalancing
fleet 270	5 min	0.68	0.68
	10 min	0.64	0.58
	15 min	0.61	0.53
fleet 360	5 min	0.64	0.63
	10 min	0.56	0.52
	15 min	0.57	0.49
fleet 450	5 min	0.59	0.58
	10 min	0.49	0.48
	15 min	0.51	0.48

Similar to the findings from above MOEs, the utilization rate is higher when the rebalancing module is effectively enhancing the system performance, which is when the updating interval length is relatively long (10 min and 15 min). There is barely any difference between the cases with and without the rebalancing module under the 5-min interval length.

One thing to note is that although the cases with the fleet size of 450 and 5-min interval length perform the best in terms of the number of served requests, the fleet utilization rate, however, is not. Such finding indicates that the high number of the served request does not necessarily lead to high usage of the fleet size. Under the same fleet size, the shorter the interval length is, the higher is the utilization rate of the MOD vehicle fleet. This is probably because the more frequent the vehicle is updated in the system, the more likely it is to achieve a vehicle-to-request match. Therefore, the vehicles are most likely to be assigned to serve a request right away when the system updates itself.

Load Factor

Besides the two measurements for MOD fleet usage above, load factor can serve as another metric to measure the fleet productivity in terms of the vehicle miles traveled. Its definition is summarized using Equation 7-3 below.

$$\text{Load Factor} = \frac{\text{(Average distance traveled/veh)* (Average No. passenger/veh)}}{\text{(Average distance traveled/veh)* (Maximum passenger capacity /veh}} \quad (7\text{-}3)$$

As presented in Table 7-13, the load factor increases with longer interval length under the same fleet size, which indicates more request-to-request matches are achieved and that the vehicles' average load is increased. Also, like all previous comparisons, the load factor is higher for cases with the rebalancing module. The improvement becomes more obvious when the rebalancing module is more functional under longer intervals.

Table 7- 13. Summary of MOD vehicle load factor- case 2

Fleet size	Interval length	With rebalancing	Without rebalancing
fleet 270	5 min	0.49	0.49
	10 min	0.63	0.63
	15 min	0.70	0.68
fleet 360	5 min	0.51	0.51
	10 min	0.65	0.64
	15 min	0.71	0.69
fleet 450	5 min	0.53	0.52
	10 min	0.65	0.64
	15 min	0.71	0.68

Other MOEs

This study further examined the impact of the rebalancing module on other measures of effectiveness, including 1) the average travel time of the requests in MOD vehicles; 2) the average travel time of the requests in metro; 3) the average total travel time of the requests; 4) the average waiting time of the requests for MOD vehicles; and 5) the average waiting time of the requests. These measurements also take the average of simulation results from three random seeds, where each random seed takes the average for all applicable MOD trips. The mean and the standard deviation of those measurements are presented in Table 7-14 and 7-16. One can see that the standard deviation is stable across all testing scenarios meaning that the performance of the system stays reliable regardless of external changes.

Table 7- 14. Summary of average request travel time - case 2

(a) Average request total travel time

Average total request travel time- MOD + metro (Hr: Min: Sec)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 270	5 min	1:19:27	0:46:26	1:19:16	0:41:26
	10 min	1:16:49	0:39:17	1:18:08	0:39:46
	15 min	1:15:07	0:38:27	1:17:24	0:38:17
fleet 360	5 min	1:19:14	0:39:33	1:19:31	0:39:43
	10 min	1:17:33	0:35:18	1:18:06	0:36:07
	15 min	1:15:25	0:33:46	1:17:18	0:35:45
fleet 450	5 min	1:19:21	0:36:46	1:19:08	0:36:39
	10 min	1:17:15	0:32:26	1:17:36	0:33:49
	15 min	1:15:45	0:29:19	1:16:48	0:35:25

(b) Average request travel time in MOD vehicle

Average request travel time- MOD (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 270	5 min	9:24	2:58	9:23	2:59
	10 min	11:31	3:04	11:16	3:00
	15 min	12:37	3:28	11:58	3:17
fleet 360	5 min	9:52	2:51	9:47	2:36
	10 min	11:49	3:03	11:57	3:02
	15 min	12:58	3:25	12:24	3:05
fleet 450	5 min	10:14	2:56	10:11	2:58
	10 min	11:51	2:59	11:57	3:02
	15 min	13:10	3:21	12:11	3:09

(c) Average request in metro travel time

Average request travel time- metro (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 270	5 min	40:17	22:26	40:30	22:26
	10 min	38:31	21:26	39:37	21:39
	15 min	37:36	21:35	39:14	21:07
fleet 360	5 min	40:17	21:57	40:26	21:36
	10 min	39:22	20:11	39:29	20:59
	15 min	38:06	19:30	39:16	20:20
fleet 450	5 min	40:11	20:17	40:11	20:32
	10 min	39:36	18:37	39:21	19:06
	15 min	38:29	17:43	39:13	20:16

The only thing to note here is that because the maximum waiting time constraint (6-minute customer walk-away limit) is only applied to the 1st sub-trip instead of being applied to both MOD trips as presented in the numerical case-1, the average customer waiting time for MOD vehicles is sometimes slightly over 5 min.

However, one surely have noticed that the average travel time for a request is over an hour in both numerical case 1 and numerical case 2. Such travel time can be longer than the direct travel time when the riders choose to drive alone. In a high-level view, this system is trading the increased travel time with less pollution and more convenience for travelers. By taking MOD service and metro, the riders don't have to drive, which brings more convience to them. To get a sense of such a trade-off, Table 7-15 below lists the increased travel time per rider and the reduced emissions brought by the system at the same time. The increased travel time per rider is the time difference between using the proposed service and driving directly without any ride-sharing. For simplicity, this study tested on the simulation setting with a fleet size equals 450 and with rebalancing logic. Such a trade-off exists in any ride-sharing systems that the riders should decide whether to sacrifice their time for lower travel cost.

Table 7- 15. Trade-off between increased travel time and reduced emission

Interval length	Overall Emission Saving (in metric tons)	Increased travel time per customer (min: seconds)
15 min	11112	22:34
10 min	9650	22:47
5 min	7401	22:55

One can see that the average travel time does increase around 20 min for a rider. However, by using such a ride-shared multi-modal service, the green gas emissions reduces significantly. From a long-run point of view, the government can promote the usage from such a system to reduce the green gas emission and protect the environment.

Table 7- 16. Summary of average request waiting time - case 2

(a) Average request metro waiting time

Average request waiting time- metro (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 270	5 min	23:09	15:43	23:00	15:45
	10 min	19:33	13:36	20:25	13:46
	15 min	17:50	13:39	19:14	13:18
fleet 360	5 min	22:45	15:17	22:48	15:15
	10 min	19:30	13:36	20:11	13:36
	15 min	17:00	12:26	18:41	12:58
fleet 450	5 min	22:28	14:07	22:18	14:18
	10 min	18:56	13:34	19:22	13:27
	15 min	17:02	12:44	18:17	12:28

(b) Average request MOD vehicle waiting time

Average request waiting time- MOD vehicles (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 270	5 min	6:07	3:38	6:08	3:28
	10 min	7:27	4:42	7:08	4:23
	15 min	7:49	4:54	7:32	4:35
fleet 360	5 min	16:16	3:39	6:19	4:02
	10 min	7:23	4:24	7:15	4:18
	15 min	8:06	4:57	7:45	4:53
fleet 450	5 min	6:28	4:02	6:28	4:01
	10 min	7:24	4:22	7:30	4:26
	15 min	8:04	4:55	7:43	4:46

In summary, those measurements are stable in terms of both their mean and standard deviation and are not impacted by the inclusion of the rebalancing module. This observation confirms the reliability of the system performance and indicates the rebalancing module can help to enhance the system efficiency when the fleet size is sufficient, and updating interval length is long enough.

Closure

To assess the effectiveness of the proposed rebalancing module and its complex interactions with the other two modules, this case study generated long-distance commuter trip patterns partially based on the demand pattern from taxicab data. To have a valid comparison between cases with and without the rebalancing module, the testing fleet size is set to be adequate, and the updating interval length is also enlarged to ensure sufficient request-to-request match opportunities.

Based on the summary results for a set of MOEs, one can clearly see that the inclusion of the rebalancing module can improve the system performance, as reflected in the increased number of served requests, more emission savings, higher average number of served requests per trip, higher fleet utilization rate and larger load factors. Such performance enhancements become more obvious under longer updating intervals because of the following two reasons. First, the longer the interval is, the more is the number of vehicles undergoing a rebalancing task that would arrive at the destination location and can be assigned to serve requests in the queue. Second, the longer interval, which in this case is 15 minutes, is a proper look-ahead window, as it gives the system enough time to prepare for the next time window and is resistant to the natural fluctuation of the demand in a short time frame.

Other MOEs, primarily measuring the service quality, showed no difference between the cases with and without the rebalancing module. Those MOEs include the average customer waiting time for MOD vehicles and metros, average customer travel time in metro, MOD vehicles and both.

Besides the two typical trade-offs seen in case -1, this case study reveals the third trade-off between the MOD fleet used for serving rider requests and rebalancing tasks, as the vehicle can only perform one task at a time. The shorter intervals lead to more chances for vehicle-to-request match opportunities. However, this only applies to vehicles not undertaking a rebalancing task. For vehicles with an on-going rebalancing task, the longer the interval length, the more likely these vehicles will arrive at the destination station and become available to serve requests. As the system planner, one should bear-in-mind this trade-off when operating the system with the rebalancing logic.

7.5. Numerical Case-3: Simulation Setting

Knowing that the synthetic data used in both previous cases fail to represent the intricacies of the real-world demand pattern fully, case-3 uses the real-world open taxi cab data to test the performance of the entire proposed system including the rebalancing module. Note that taxi trips are relatively shorter compared with long comminuting trips generated in the previous two cases. The results of case-3 validate the system performance under a demand pattern with short-distance trips. Besides the pattern change, the author further compares the effectiveness of the rebalancing module under such demand pattern.

Figure 6-3 shows the selected nine stations according to the taxi data coordinates. This data input applies one day of the taxicab demand used in the demand forecasting module so that the station selections are the same. Note that this case study uses rectilinear distance to generate the results for the map-partitioning module.

These nine stations are selected when a balance between the in-cluster sparsity and between cluster demand sparsity is observed. Such a balance is typically used to evaluate the goodness of the clustering algorithm. The detailed explanation of the reason for these selections is presented in chapter 6.

This numerical test was carried using the real-data from 6:00 am to 6:00 pm on July 27th, 2017. The predicted hourly demand is generated by the convolutional neural network, which is presented in chapter 6. When the simulation is running on different interval length, the hourly demand forecasting is adjusted accordingly. For example, if the CNN predicts that in one of the stations there are 100 pickups in the next hour, then there will be 25 vehicles per 15 minutes for the target station if the interval length is 15 min.

Similar to case-1 and 2, the MOD vehicle's initial location is randomly generated across the map. Three randomly generated seeds are used to avoid the potential bias caused by one random seed. However, unlike the previous 2 cases where the requests reply on the random generation, the initial location of the MOD vehicles only affect the beginning of several iterations. After the initial vehicle-to-station assignment is complete, the three cases generate almost the same results as the vehicles will stabilize to respond to the same demand pattern. The summary results of the selected MOEs are the average of all three random seeds. Note that this case study sets the same MOE as the one in case 2.

In this numerical study we also perform sensitivity analysis with a set of varying key control parameters, summarized as follows:

- Total MOD vehicle fleet size: 90, 135, 180, 225, and 270;
- Length of the iteration time window: 5 min, 10 min, and 15 min;
- Simulation Time and duration: 06:00 am to 6:00 pm on July 27th, 2017.
- MOD vehicle maximum capacity: 3 passengers excluding the driver.

The simulation program is coded in *Python 3.4,* and *Gurobi* optimizer is used as the solver for the optimization model. Due to the unavailability of past metro schedule from the WMATA API, the simulation was performed based on a Thursday schedule of 2019 (January 30th, 2019) instead, as the simulated data is also from a regular Thursday. The MOD vehicles are assumed to be homogeneous regular passenger cars with a maximum capacity of three passengers, excluding the driver.

The real-time simulations were then conducted based on three types of updating intervals: 5-min, 10-min, and 15-min. Note that the interval length is relatively long compared to the numerical case-1 because the magnitude of the taxi-data is much less than the simulated demand in case-1. Longer interval is necessary to ensure sufficient requests are received to conduct ride-sharing tests. The waiting time constraints were applied to the 1st sub-trip for all requests. This is due to the same reason as presented in case 2.

If the coordinates of the requested pickup or drop-off locations are within the 0.5 miles of the stations, then it is assumed that the customer will walk to the station instead of riding the MOD vehicles. In this case, the time feasibility check will be conducted based on the average human walking speed of 3.1 miles per hour. All the other settings are the same as the previous two case studies.

7.6. Numerical Case-3: Simulation Results

This section selects a set of performance measures, including those presented in case-1. Note that to test the potential benefits of implementing the rebalancing module, this simulation assumes 100% compliance rate for MOD vehicles to complete the assigned relocation works. The following presentations of the MOEs are based on the aggregate results of all applicable sub-trips.

Number of Severed Requests

Figure 7-8 shows the comparative results of the number of served requests with respect to different fleet sizes, updating interval lengths. For each key control parameter combination, cases with and without the rebalancing modules are tested. Similar to the observation in case-1 and case-2, under a given demand, the number of served requests increases with larger total vehicle fleet size.

In contrast to the previous two cases, the fleet size in this case study is set to be smaller than the previous two cases. This is because we also want to see how the fleet size will affect the effectiveness of the rebalancing module, as this was tested in case 2. The simulation results show the number of served requests increases when the available fleet size increases. The number of served requests in cases with or without the rebalancing module are similar regardless of varying interval lengths and when the fleet size is at its smallest level, which is equal to 90. This indicates that the service quality is mainly constrained by the insufficient supply of MOD vehicles.

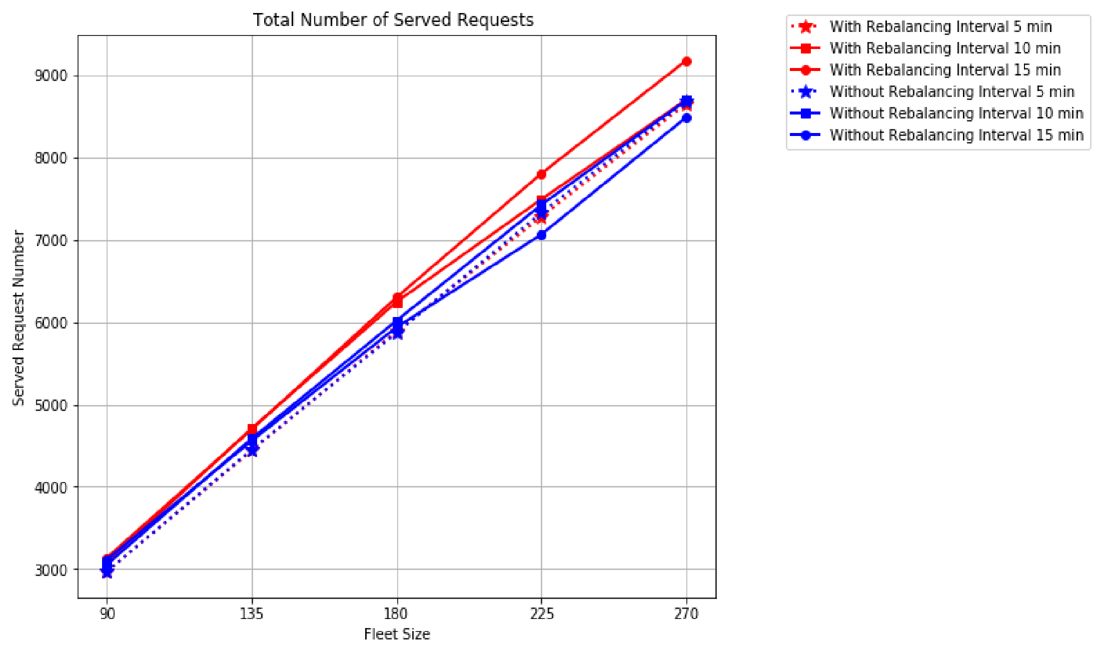


Figure 7- 8. Number of served requests w/w.o rebalancing module- case 3

The increase in the number of served requests in cases with the rebalancing becomes more obvious with longer updating interval length and the larger fleet size. This finding is the same as the one made in case-2, where the rebalancing module gives more performance enhancement when interval length is reasonably large.

This might be because the 15-minute interval length is the best rebalancing interval length among all three tested interval durations. As indicated in the development of the rebalancing module, the rebalancing interval length needs to be tested because too small an interval length ignores the nature of demand fluctuations and causes unnecessary vehicle relocation while one that is too large limits the system's ability to react to the incoming demand. Moreover, the longer interval also means more vehicles with the rebalancing task become available within the iteration, so that could serve more requests. Under the test demand scenarios in case-3, the case with a fleet size of 270, 15-minute interval length and with the rebalancing module serves the most requests.

In conclusion, with the addition of the rebalancing module, the system can reliably serve more requests compared with the case without rebalancing. Its benefit will be more obvious under sufficient fleet size and interval length, as shown in case 2. However, even when the fleet size is not sufficient, the rebalancing module will not negatively affect system efficiency. Also, in terms of the number of served requests, there is an apparent complex interaction among all key control parameters, including fleet size, demand level, and updating interval length. For real-world planning purposes, the best combination of the interval length and fleet size should be tested under a given demand level to achieve the optimal system performance.

Total Emission Saving

Similar to the previous two cases, this case study also applies Equation 7-1 to calculate the reduction of greenhouse gas emission. The definition of the terms is the same as those presented in case 2. However, because the requests are short-distance trips, the emission saving is much less obvious compared with the previous two cases.

For better visualization purpose, Figure 7-9 measures emission saving in metric kilograms instead of metric tons as presented in case-1 and case-2. Compared with the promising emission saving in case-1, this case does not show much environmental benefits in reducing greenhouse gas emissions. This is pretty much because the taxicab trips are relatively shorter in the distance as compared with the simulated long-distance trips in case-1 and case-2. This reduction makes sense as there is less mileage saving for using the proposed system as compared to the direct travel when trips are mostly of short-distance.

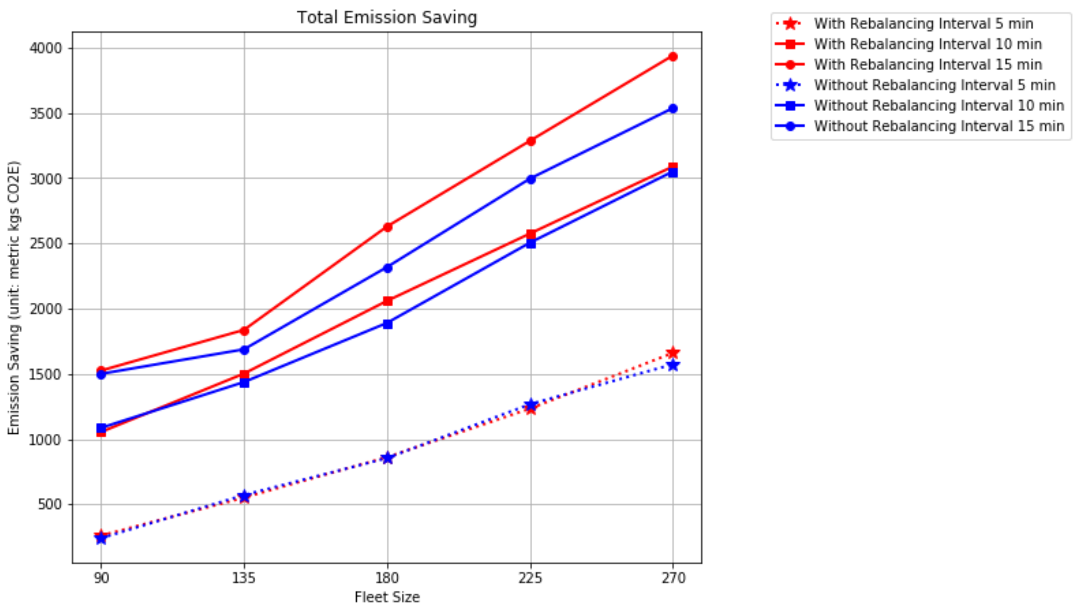


Figure 7- 9. Total emission saving w/w.o rebalancing module

Although the emission savings are much less compared to long commuter trips, one can see from the above figure that all emission savings are strictly positive meaning that the vehicle relocation does not cause unnecessary empty miles, which could lead to negative emission savings. The inclusion of the rebalancing module in the worst case scenario when there is no vehicle to be relocated would perform just as the same as the cases without rebalancing.

Although the emission saving benefit is compromised under short-distanced demand, the parking cost saving should be promising. The average daily parking cost in DC is $25 per vehicle per day. If we assume that all trips are commuters, the parking cost saving can range from 0.02 to 0.07 million dollars per day in total. That would also count as a benefit brought about by this system.

Average Number of Served Requests per Trip

Table 7-17 summarizes the average number of served requests per trip as well as its standard deviation given different fleet size and interval length combinations. Similar to the findings from analyzing the number of served requests, the average number of served requests per trip is increasing with the longer interval length under the same fleet size. Under the same fleet size and interval length, this term is slightly higher with the rebalancing module compared to the case without it. However, such improvement by the rebalancing module is less obvious under a small fleet size. Note that due to the maximum passenger capacity which is three, the closer the average number of served requests per trip is to 3, the better is the system performance.

Table 7- 17. Summary of the average no. of served requests per trip-case 3

Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 90	5 min	1.33	0.58	1.33	0.57
	10 min	1.68	0.81	1.68	0.81
	15 min	1.99	0.97	1.96	0.97
fleet 135	5 min	1.36	0.67	1.36	0.67
	10 min	1.78	0.92	1.73	0.91
	15 min	2.02	1.04	1.99	1.04
fleet 180	5 min	1.38	0.74	1.38	0.74
	10 min	1.81	0.97	1.76	0.96
	15 min	2.11	1.1	2.04	1.08
fleet 225	5 min	1.41	0.78	1.41	0.79
	10 min	1.79	0.97	1.79	0.98
	15 min	2.14	1.1	2.07	1.08
fleet 270	5 min	1.43	0.81	1.43	0.81
	10 min	1.84	1	1.83	0.99
	15 min	2.15	1.09	2.1	1.08

Utilization Rate

Similar to case-2, this case study also employs the utilization rate to assess fleet usage. Its calculation method is presented in Equation 7-2.

Similar to the observations made in the number of served requests, the utilization rate, as summarized in table 7-18, is relatively higher when the rebalancing module is in effect; otherwise, it is the same as the test cases without rebalancing logic.

Under the same fleet size, the shorter the interval length, the higher the utilization rate of the MOD vehicle fleet. This is probably because the more frequent the vehicle is updated in the system, the more likely it is to achieve a vehicle-to-request match. Therefore, the vehicles are most likely to be assigned to serve a request right away when the system updates itself. However, the higher utilization rate does not necessarily lead to an increased number of served requests, as the opposite trends are observed from the analysis of the number of served requests. Such an opposite trend, again, justifies the two trade-offs between vehicle-to-request and request-to-request match cases, as indicated in numerical case-1.

Table 7- 18. Summary of MOD vehicle utilization rate - case 3

Fleet size	Interval length	With rebalancing	Without rebalancing
fleet 90	5 min	0.79	0.78
	10 min	0.74	0.73
	15 min	0.70	0.70
fleet 135	5 min	0.78	0.78
	10 min	0.74	0.72
	15 min	0.69	0.68
fleet 180	5 min	0.77	0.76
	10 min	0.72	0.71
	15 min	0.68	0.66
fleet 225	5 min	0.76	0.76
	10 min	0.70	0.69
	15 min	0.66	0.64
fleet 270	5 min	0.75	0.74
	10 min	0.67	0.67
	15 min	0.65	0.61

Load Factor

Table 7-19 summarizes the load factors, which is calculated using equation 7-3. Like the observed trend in the average number of served requests per trip, the load factor is increasing with the increase in interval length under the same fleet size, which indicates more request-to-request matches are achieved and that the vehicle's average load is increased. Also, comparing the cases with and without the rebalancing module, the ones with the module either generate slightly higher load factor or perform almost the same as the ones without it, with and without sufficient fleet size. The rebalancing logic is effective when fleet size is sufficient.

Table 7- 19. Summary of MOD vehicle load factor- case 3

Fleet size	Interval length	With rebalancing	Without rebalancing
fleet 90	5 min	0.44	0.44
	10 min	0.56	0.56
	15 min	0.66	0.65
fleet 135	5 min	0.45	0.45
	10 min	0.59	0.58
	15 min	0.67	0.66
fleet 180	5 min	0.46	0.46
	10 min	0.60	0.59
	15 min	0.70	0.68
fleet 225	5 min	0.47	0.47
	10 min	0.60	0.60
	15 min	0.71	0.69
fleet 270	5 min	0.48	0.48
	10 min	0.61	0.61
	15 min	0.72	0.70

Other MOEs

This study further examines the impact of the rebalancing module on other measures of effectiveness, including 1) the average travel time of the requests in MOD vehicles; 2) the average travel time of the requests in metro; 3) the average total travel time of the requests; 4) the average waiting time of the requests for MOD vehicles; and 5) the average waiting time of the requests. The mean and the standard deviation of those measurements are presented in the following table 7-20 and 7-21.

Table 7- 20. Summary of average request travel time- case 3

(a) Average request total travel time

Average total request travel time- MOD + metro (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 90	5 min	49:08	19:57	49:42	21:07
	10 min	48:00	21:19	48:02	21:26
	15 min	48:40	22:09	49:13	22:50
fleet 135	5 min	45:22	20:35	45:05	20:23
	10 min	46:07	21:09	46:34	22:22
	15 min	46:22	22:44	47:03	22:19
fleet 180	5 min	44:10	21:17	43:52	21:11
	10 min	44:31	21:13	45:10	22:13
	15 min	44:34	22:13	45:32	22:44
fleet 225	5 min	43:21	21:39	43:18	21:41
	10 min	44:30	21:13	44:22	21:13
	15 min	42:56	21:35	44:42	21:24
fleet 270	5 min	42:36	21:25	42:27	21:20
	10 min	43:32	20:58	43:38	21:04
	15 min	42:58	20:59	43:57	21:22

(b) Average request travel time in MOD vehicles

Average Request Travel time- MOD (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 90	5 min	7:05	2:17	7:06	2:17
	10 min	8:31	3:07	8:36	3:14
	15 min	9:31	4:27	9:32	4:34
fleet 135	5 min	7:12	3:22	7:11	3:22
	10 min	8:53	4:20	8:33	4:13
	15 min	9:36	4:36	9:30	4:06
fleet 180	5 min	7:19	3:54	7:18	3:54
	10 min	8:45	4:35	8:38	4:32
	15 min	9:42	4:47	9:31	4:17
fleet 225	5 min	7:21	3:15	7:29	3:08
	10 min	8:50	4:45	8:42	4:42
	15 min	9:37	4:18	9:36	4:18
fleet 270	5 min	7:26	3:28	7:25	3:31
	10 min	8:48	4:30	8:45	4:51
	15 min	9:38	4:17	9:30	4:07

(c) Average Request Travel Time in Metro

Average Request Travel time- Metro (minutes: seconds)					
Fleet Size	Interval Length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 90	5 min	14:12	6:16	14:08	6:58
	10 min	13:50	7:39	14:05	7:50
	15 min	14:04	7:57	14:14	7:58
fleet 135	5 min	13:32	7:34	13:36	7:35
	10 min	12:57	7:58	13:22	8:03
	15 min	12:43	8:08	13:22	8:23
fleet 180	5 min	13:18	7:58	13:12	7:58
	10 min	12:27	8:18	12:41	8:25
	15 min	12:01	8:18	12:40	8:38
fleet 225	5 min	12:53	8:10	12:54	8:11
	10 min	12:16	8:35	12:21	8:36
	15 min	11:51	8:31	12:14	8:39
fleet 270	5 min	12:37	8:22	12:40	8:25
	10 min	12:06	8:33	12:06	8:34
	15 min	11:52	8:25	11:57	8:34

The only thing to note here is that because the maximum waiting time constraint (6-minute customer walk-away limit) is only applied to the 1st sub-trip instead of being applied to both MOD trips the average customer waiting time for MOD vehicles is sometimes slightly over 5 min. Also, one can see that for all of the listed MOEs, their standard deviation are similar across all the testing scenarios. This indicates that the system can deliver stable performance regardless of the varying key control parameter settings. Such stability also proves the reliability of the proposed system as a whole.

Table 7- 21. Summary of average request waiting time- case 3

(a) Average request waiting time for MOD vehicles

Average request waiting time- MOD vehicles (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 90	5 min	4:31	2:18	4:27	2:12
	10 min	5:25	2:41	5:21	2:46
	15 min	5:58	2:59	5:53	2:47
fleet 135	5 min	4:35	2:11	4:31	2:08
	10 min	5:32	2:50	5:20	2:38
	15 min	5:52	3:00	5:52	2:58
fleet 180	5 min	4:32	2:18	4:32	2:21
	10 min	5:54	2:53	5:50	2:56
	15 min	5:52	2:58	5:50	2:58
fleet 225	5 min	4:38	2:36	4:36	2:34
	10 min	5:29	2:41	5:21	2:43
	15 min	5:42	2:59	5:51	3:03
fleet 270	5 min	4:42	2:29	4:36	2:28
	10 min	5:27	2:49	5:23	2:47
	15 min	5:42	2:52	5:48	2:58

(b) Average request waiting time for metro

Average request waiting time- metro (minutes: seconds)					
Fleet size	Interval length	With rebalancing		Without rebalancing	
		Mean	S.D.	Mean	S.D.
fleet 90	5 min	13:10	6:57	13:55	7:05
	10 min	8:41	4:20	8:33	4:30
	15 min	6:49	3:12	7:18	3:54
fleet 135	5 min	9:38	4:56	9:24	4:58
	10 min	6:53	3:28	7:42	3:58
	15 min	5:43	2:43	5:55	3:03
fleet 180	5 min	8:30	4:15	8:18	4:12
	10 min	6:00	3:01	6:39	3:24
	15 min	4:31	2:16	5:05	2:38
fleet 225	5 min	7:45	3:57	7:38	4:01
	10 min	5:55	2:46	6:04	2:59
	15 min	3:28	1:37	4:32	1:54
fleet 270	5 min	6:55	4:58	7:00	5:00
	10 min	5:05	4:17	5:15	4:25
	15 min	3:28	4:02	4:11	4:18

In summary, those measurements are stable in terms of both their mean and standard deviation and are not impacted by the inclusion of the rebalancing module, varying levels of fleet size, and interval lengths. This observation confirms the reliability of the system performance and indicates the rebalancing module does not negatively affect other system performance metrics even when the trips are primarily short-distance.

Closure

To capture the real-world traffic pattern imbalance, in numerical case-2 we imported the DC taxicab data as the input demand and assessed the performance of the proposed system with and without the rebalancing module. As presented in Chapter 6, the station-based demand forecasting results generated by CNN were used in the simulation cases with the rebalancing module. Three random seeds were selected to generate random initial locations for the MOD vehicle of different total fleet size to ensure the results reliability. For each random seed, a total of 12 hours simulation with three different updating intervals were performed, and a set of system performance metrics were collected. Besides the same MOEs presented in numerical case -1, this section together with case 2 also presented some additional MOEs, such as utilization rate and load factors, that would help to provide a more comprehensive understanding of the fleet utilization and system efficiency.

As indicated by the extensive simulation results, the integration of the rebalancing module with the other two modules outperforms the system without the rebalancing logic if the available fleet size is sufficient and proper interval length is chosen. Such a benefit is confirmed by both demands with long-commuter trips and short-distanced taxicab trip. However, even without setting these two parameters properly, the inclusion of the rebalancing module does not negatively affect the system functionality, rather it produces almost the same performance as the system without the rebalancing module.

Two typical trade-offs, as observed in numerical case-1 and case -2, also were observed here. This finding further confirms the complex interactions among the key control parameters and how sensitive the overall system performance would react to their changes. The optimal combination of the key control parameters, such as the interval length for ride-sharing and the rebalancing modules, and total fleet size, can be determined by simulation experiments in the planning phase.

7.7. Summary

This chapter presented three numerical case studies. Numerical case-1 generated hypothetical daily commuter trips under high demand to assess the reliability and robustness of the first two modules, map-partitioning, and ride-sharing. Based upon the understanding of the working mechanism of the 1st two modules, case study 2 further tested the potential benefit brought by the rebalancing module with long-commuter demand pattern. With sufficient fleet size, the rebalancing module can evidently enhance the system efficiency while maintaining the same level of service quality. To test the system's performance over real-world demand data, the case 3 employs DC taxicab data to test how it would react to the demand with short-distance and specifically, how would the rebalancing module be affected by that.

A collection of performance measures that characterize the system efficiency and service quality were recorded. Those measures include the number of served requests, total emission saving, the average number of served requests per trip, the average customer waiting time for the MOD vehicles and so on. In case-1, the results of extensive simulation runs prove the applicability of the proposed system and justify its immense potential for pollution reduction. With the plot of the total number of served requests, one can select the best combination of the total fleet size and the interval length under a given demand level. This helps the system operator decide the appropriate fleet size to maintain so that they can stimulate more drivers into the system or assign them to other services by adjusting the price.

There are two main trade-offs. One is reflected in the selection of interval length. Shorter interval lengths generate more opportunities for vehicle-to-request matches while longer ones provide more ride-sharing possibilities. The service provider has to find an interval length that balances the odds for both matches such that the system can perform most efficiently. Generally, when both demand and supply are high, lower interval lengths performs the best.

The other trade-off is reflected via the customer waiting time constraint. Imposing this constraint can significantly reduce the customer waiting time for MOD vehicles under their common tolerance. Hence, the service quality (for the served requests) is improved. However, this constraint influences the total number of served requests negatively, forcing the system to decline many requests that can be served without violating their arrival time window. Such unfavorable influence was also reflected in the decreased emission saving and the average served requests per trip. This helps the decision makers to recognize the trade-off between service quality and system efficiency. One can also resort to pricing strategies to encourage riders to tolerate a longer waiting time to enhance ride-sharing opportunities.

Numerical case-2, with the inclusion of rebalancing module, also detects the third trade-off. This trade-off also reflects upon the selection of updating interval length as the shorter interval enables more potential for an available vehicle-to-request match. However, the longer interval could provide more chance for vehicles with rebalancing tasks to arrive at the destination station, thus adding more available fleet size into the vehicle pool. Note that this trade-off is, essentially, the balance between vehicles in-service and vehicles in relocation.

Numerical case-3 used real-world taxi data to represent the unevenness of the demand distributions across the map with the main goal of testing how the integration of all three modules would react to the demand with short-distance trips. Similar to the case with long-commuter trips, the inclusion of the rebalancing module can increase system efficiency by serving more requests while maintaining the same level of service quality compared to the cases without rebalancing, as indicated by waiting time, travel time and so on. The mean and the standard deviation of the service quality measures don’t change if the rebalancing module doesn’t improve the number of served request. That is to say; the rebalancing module does not affect the system performance negatively under any circumstances.

Both in case 2 and case 3, three different looking-ahead windows for rebalancing modules were tested. Performance discrepancies were observed across different window lengths, and one of three consistently outperforms the other two under all fleet sizes. This does confirm the look-ahead window length is a key parameter to ensure the efficiency of vehicle relocations. With the testing demand pattern, the longer interval length enables better performance of the rebalancing module. This further confirms the system’s reliability and contribution under medium demand for shorter trips.

This study, in summary, provided some initial insights into how to integrate the MOD and mass transit system seamlessly. Simulation results revealed the complex interactions among the key control parameters, including interval lengths and fleet size. At the planning phase, one should attempt to optimize them concurrently under a certain demand level and bear in mind the key trade-offs between system efficiency and service quality.

Chapter 8: Conclusions and Future Work

8.1. Research Summary and Contribution

This study presented a multi-modal ride-share system to serve as an environmentally friendly alternative to private car usage. As long been promoted by governments globally, the most efficient and scalable urban transportation model is via mass transit, such as metro lines, buses, etc. However, the utilization of these public systems is limited due to the lack of first-and-last mile solutions. The MOD system possesses great potential to bridge this gap. By integrating with the mass transit system, the MOD vehicles can reduce detours and polluting emissions. Acknowledging that the commuter traffic may share the route to the stations and to operate this system at a viable price point, the first and last mile trips are most likely to be a shared-ride. Therefore, this study incorporated but is not limited to, the ride-sharing service for the first and last mile trips.

This study proposed a framework for assessing the potentials of an integrated MOD –mass transit system and realized it through extensive simulation experiments. A recursive optimal method was proposed to solve the vehicle-to-rider and the rider-to-rider matches with scalability concurrently. The algorithm consists of three key modules which are map-partitioning, ride-sharing and rebalancing modules. The inputs include request information, vehicle status, and the metro schedule. The map partitioning module first clusters the large-scale demand and divides the entire map into smaller zones to reduce the computational expense. This information can later be inputted into the ride-sharing model, which can generate an optimized vehicle-to-request and request-to-request matches in a timely fashion. The module employs Request-Schedule-Vehicle graph to check the trip feasibility and then solves the optimal ride-sharing plan using a MIP. The last module, rebalancing, is used to relocate the vehicles in preparation for the incoming demands. Embedded in this module, a scalable demand forecasting model is developed to produce hourly pickup and drop-off prediction for each station. This study used the open DC taxi data to develop three station-based demand forecasting models and selected the one with a balance between computational efficiency and prediction accuracy. The forecasting results were then inputted into an optimization model that seeks to serve as many incoming requests as possible by relocating vehicles in advance.

Three numerical case studies were presented to test the system's reliability and robustness. Both long-distance and short-distance demand patterns with varying magnitude were examined to unveil the system's working mechanism. The simulation results justified the proposed system can serve a large number of requests in a limited time frame with desired service quality. Compared with the short-distance demands, the system could deliver much higher benefit when most of the requests are long-commuting traffic. This matches the expectation as the long-distance travel make more sense to use multiple transportation modes. The results quantify experimentally two trade-offs that reflect on the customer waiting time, fleet size, and the updating interval length. The simulation results from the last two case studies justify that the rebalancing module can enhance the system efficiency with a desired service quality. In the planning phase, the system planner can decide the best combination of fleet size and interval length to achieve ideal performance. A third trade-off exists between the available fleet size used for serving rider request and conducting rebalancing task, respectively. The simulation results from all numerical cases show a clear benefit for using such an integrated system, in particular, with respect to the reduction in vehicular emissions and the parking cost saving as compared with the private car mode.

The potential provider of the service is not limited to the typical MOD service provider, like Uber or Lyft, but can also be a metro company that hopes to boost its user populations. For example, Uber can provide such a service to reduce MOD vehicle re-routing since each vehicle is assigned to a certain station and they can further apply smart pricing strategies to stimulate or reduce the requests. Metro systems can use the service as the first-and-last mile solution to enhance public transportation usage. This strategy can also be used by a third party who coordinates both sides.

In Summary, this study has made the following key contributions:

- Designing a seamlessly integrated multi-modal ride-sharing system which allows the MOD service to function as a feeder to the under-used public transit system. This system can increase vehicle occupancy rate and reduce vehicular emission.
- Proposing a real-time updating process that enables the proposed methodology to solve large-scale ride-sharing problems in a timely fashion. This is the key to real-world application.

- Constructing a Request-Schedule-Vehicle graph and formulating the mixed-integer programming models to solve the optimal rider to vehicle and vehicle to metro schedule matches concurrently. This method also ensures the three components of any rider trips be solved with integrity.
- Designing an efficient fleet management strategy that could redistribute the MOD vehicle supply to accommodate the rider request dynamics. Embedded in this module, is a scalable and accurate demand forecasting model that assists the planner to prepare for the incoming demand.

8.2. Potential Future Research

Despite the progress made in the study, much remains to be done to address the complexities in operating this multi-modal MOD system. The contributions of this research lie primarily in the planning phase of such a system design. However, there are some necessary extensions to push it into the real-world application domain:

1. **Development of Smart Pricing Strategies**: As reported by Uber, the demand and supply within the MOD system tend to react sensitively to its pricing. For application purposes, a smart and dynamic pricing tool is a necessity to ensure its success. It comes into play in the following two aspects: 1) dynamically adjusting the demand and supply, and providing more profit for the driver when the system is short of MOD vehicles; 2) increasing the compliance rate for vehicles that follow the rebalancing task orders. In reality, the MOD driver would reject a rebalancing order if he or

she cannot see a direct profit. A smart pricing strategy would stimulate them to comply with the assigned relocation work.

2. **Testing on Dynamic MOD Supplies**: This study, so far, is testing the system performance based on fixed fleet size. However, with the embedded recursion feature, the vehicle information pool can be updated to reflect the change in MOD fleet size at the beginning of each iteration. Thus, making updating the total fleet size possible. Future work aligns with testing the system performance and reliability with a relaxation of fixed MOD fleet size.
3. **Rebalancing Task Compliance Rate**: Under a real-world scenario, the MOD drivers can sometimes be reluctant to relocate from one station to the other, especially when traveling with no passenger. Besides paying them to do so, the system can also take advantage of assigning them real requests whose destinations are in the target relocation zone. To achieve this, the ride-sharing and rebalancing modules must be modified accordingly to allow cross zone traveling. A binary compliance indicator should also be introduced to update the vehicle status.
4. **Varying Simulation Settings**: One can further test the system performance with varying MOD vehicle capacity, different urban area layouts, and levels of compliance for MOD driver to take the rebalancing order.

References

Agatz, N. A., Erera, A. L., Savelsbergh, M. W., & Wang, X. (2011). Dynamic ride-sharing: A simulation study in metro Atlanta. Transportation Research Part B: Methodological, 45(9), 1450-1464.

Agatz, N., Erera, A., Savelsbergh, M. and Wang, X., 2012. Optimization for dynamic ride-sharing: A review. European Journal of Operational Research, 223(2), pp.295-303.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., & Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proceedings of the National Academy of Sciences, 114(3), 462-467.

Baldacci, R., Maniezzo, V., and Mingozzi, A., 2004. An exact method for the car pooling problem based on lagrangean column generation. Operations Research, 52(3), pp.422-439.

Barrios, J., On the performance of flexible carsharing: a simulation-based approach. j, 2012.

Barth, M. and M. Todd, Simulation model performance analysis of a multiple shared vehicle system. Transportation Research Record, 1999.

Berbeglia, G., Cordeau, J. F., & Laporte, G. (2010). Dynamic pickup and delivery problems. European journal of operational research, 202(1), 8-15.

Bezdek, J. C. (1981). Objective Function Clustering. In Pattern recognition with fuzzy objective function algorithms (pp. 43-93). Springer, Boston, MA.

Bezdek, J.C., 1973. Fuzzy mathematics in pattern classification. Ph. D. Thesis, Applied Math. Center, Cornell University.

Carrion C, Levinson D (2012) Value of travel time reliability: a review of current evidence. Transp Res Part A Policy Pract 46(4):720–741.

Chen, P.Y., Liu, J.W. and Chen, W.T., 2010, September. A fuel-saving and pollution-reducing dynamic taxi-sharing protocol in VANETs. In Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd (pp. 1-5). IEEE.

de Almeida Correia, G.H., and Antunes, A.P., 2012. Optimization approach to depot location and trip selection in one-way carsharing systems. Transportation Research Part E: Logistics and Transportation Review, 48(1), pp.233-247.

Di Febbraro, A., Gattorna, E., and Sacco, N., 2013. Optimization of dynamic ridesharing systems. Transportation Research Record: Journal of the Transportation Research Board, (2359), pp.44-50.

d'Orey, P.M., Fernandes, R. and Ferreira, M., 2012, September. Empirical evaluation of a dynamic and distributed taxi-sharing system. In Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on (pp. 140-146). IEEE.

Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters.

Efthymiou, D., C. Antoniou, and Y. Tyrinopoulos, Spatially Aware Model for Optimal Site Selection: method and Application in a Greek Mobility Center. Transportation Research Record, 2012.

EPA (2017). Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990-2015. Chapter 3 (Energy), Tables 3-12, 3-13, and 3-14. Environmental Protection Agency, Washington, D.C. EPA #430-P-17-001.

European Environment Agency, 2010. Occupancy rates of passenger vehicles. Tech. rep., European Environment Agency.

Fadili, M.J., Ruan, S., Bloyet, D. and Mazoyer, B., 2001. On the number of clusters and the fuzziness index for unsupervised FCA application to BOLD fMRI time series. Medical Image Analysis, 5(1), pp.55-67.

Fagnant, D.J., and Kockelman, K.M., 2014. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. Transportation Research Part C: Emerging Technologies, 40, pp.1-13.

Fagnant, D.J., and Kockelman, K.M., 2018. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, Texas. Transportation, 45(1), pp.143-158.

Fan, W., Machemehl, R., and Lownes, N., 2008. Carsharing: Dynamic decision-making problem for vehicle allocation. Transportation Research Record: Journal of the Transportation Research Board, (2063), pp.97-104.

FHWA (2017). Highway Statistics 2015. Office of Highway Policy Information, Federal Highway Administration. Table VM-1.

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.E., Wang, X. and Koenig, S., 2013. Ridesharing: The state-of-the-art and future directions. Transportation Research Part B: Methodological, 57, pp.28-46.

Geisberger, R., Luxen, D., Neubauer, S., Sanders, P. and Volker, L., 2009. Fast detour computation for ride sharing. arXiv preprint arXiv:0907.5269.

Ghoseiri, K., 2012. Dynamic rideshare optimized matching problem (Doctoral dissertation).

Gidofalvi, G. and Pedersen, T.B., 2007. Cab-sharing: An effective, door-to-door, on-demand transportation service. In The 6th European Congress on Intelligent Transport Systems and Services, 18-20 June 2007, Aalborg, Denmark (p. 8). ERTICO.

Gonzalez, H., Han, J., Li, X., Myslinska, M., and Sondag, J.P., 2007, September. Adaptive fastest path computation on a road network: a traffic mining approach. In Proceedings of the 33rd international conference on Very large data bases (pp. 794-805). VLDB Endowment.

Gurobi Optimization, I. (2015). Gurobi optimizer reference manual. URL http://www. gurobi. com.

Hathaway, R.J. and Bezdek, J.C., 1993. Switching regression models and fuzzy clustering. IEEE Transactions on fuzzy systems, 1(3), pp.195-204.

Hennessy DA, Wiesenthal DL (1999) Traffic congestion, driver stress, and driver aggression. Aggress Behav 25(6):409–423.

Horn, M. E. (2002). Fleet scheduling and dispatching for demand-responsive passenger services. Transportation Research Part C: Emerging Technologies, 10(1), 35-63.

Hruschka, H., 1986. Market definition and segmentation using fuzzy clustering methods. International Journal of research in Marketing, 3(2), pp.117-134.

https://climateprotection.org/carma-carpooling/

https://developer.wmata.com/

https://developers.google.com/maps/documentation/distance-matrix/policies

https://flinc.org/

https://www.lyft.com/line

https://www.side.cr/

https://www.uber.com/en-SG/drive/singapore/resources/uberpool/
http://opendata.dc.gov/datasets/taxicab-trips-sampling-in-july-2017

Jaw, J. J., Odoni, A. R., Psaraftis, H. N., & Wilson, N. H. (1986). A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transportation Research Part B: Methodological, 20(3), 243-257.

Jorge, D., Correia, G.H., and Barnhart, C., 2014. Comparing optimal relocation operations with simulated relocation policies in one-way carsharing systems. IEEE Transactions on Intelligent Transportation Systems, 15(4), pp.1667-1675.

Karypis, G. and Kumar, V., 1995. METIS--unstructured graph partitioning and sparse matrix ordering system, version 2.0.

Karypis, G. and Kumar, V., 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 20(1), pp.359-392.

Kek, A., R. Cheu, and M. Chor, Relocation Simulation Model for multiple-station shared-use vehicle systems. Transportation Research Record, 2006.

Koffman, D., 2004. Operational experiences with flexible transit services. No. 53. Transportation Research Board.

Lee, A. and Savelsbergh, M., 2017. An extended demand responsive connector. EURO Journal on Transportation and Logistics, 6(1), pp.25-50.

Li, X., Quadrifoglio, L., 2010. Feeder transit services: Choosing between fixed and demand responsive policy. Transportation Research Part C: Emerging Technologies 18 (5), 770 – 780.

Li, X., Quadrifoglio, L., 2011. 2-vehicle zone optimal design for feeder transit services. Public Transport 3 (1), 89–104.

Ma, S., Zheng, Y., & Wolfson, O. (2013, April). T-share: A large-scale dynamic taxi ridesharing service. In Data Engineering (ICDE), 2013 IEEE 29th International Conference on (pp. 410-421). IEEE.

McKenzie, B., October 2010. Public transportation usage among U.S. workers: 2008 and 2009. Tech. Rep., U.S. Census Bureau.

Mes, M., Van Der Heijden, M., and Van Harten, A., 2007. Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. European journal of operational research, 181(1), pp.59-75.

Mitrović-Minić, S. and Laporte, G., 2004. Waiting strategies for the dynamic pickup and delivery problem with time windows. Transportation Research Part B: Methodological, 38(7), pp.635-655.

Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., and Willhalm, T., 2005, May. Partitioning graphs to speed up Dijkstra's algorithm. In International Workshop on Experimental and Efficient Algorithms (pp. 189-202). Springer, Berlin, Heidelberg.

Myhrvold, C., Uber Expectations As We Grow, 2015.

Nalepa, J., & Blocho, M. (2015, November). A parallel algorithm with the search space partition for the pickup and delivery with time windows. In P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015 10th International Conference on (pp. 92-99). IEEE.

Ozkan, I. and Turksen, I.B., 2007. Upper and lower values for the level of fuzziness in FCM. In Fuzzy Logic (pp. 99-112). Springer, Berlin, Heidelberg.

Pal, N.R. and Bezdek, J.C., 1995. On cluster validity for the fuzzy c-means model. IEEE Transactions on Fuzzy Systems, 3(3), pp.370-379.

Pant P, Harrison RM (2013) Estimation of the contribution of road traffic emissions to particulate matter concentrations from field measurements: a review. Atmos Environ 77:78–97.

Papanikolaou, D., A New System Dynamics Framework for Modelling behavior of vehicle sharing systems. Proc. of Symposium on Simulation for Architecture and urban design, 2011.

Pavone, M., Smith, S.L., Frazzoli, E., and Rus, D., 2012. Robotic load balancing for mobility-on-demand systems. The International Journal of Robotics Research, 31(7), pp.839-854.

Pelzer, D., Xiao, J., Zehe, D., Lees, M.H., Knoll, A.C. and Aydt, H., 2015. A partition-based match making algorithm for dynamic ridesharing. IEEE Transactions on Intelligent Transportation Systems, 16(5), pp.2587-2598.

Polzin, S., Pisarski, A., April 2013. Commuting in Aamerica 2013: The national report on commuting patterns and trends.

Qiu, F., Li, W., & Haghani, A. (2015). A methodology for choosing between fixed-route and flex-route policies for transit services. Journal of Advanced Transportation, 49(3), 496-509.

Sáez, D., Cortés, C.E., and Núñez, A., 2008. Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. Computers & Operations Research, 35(11), pp.3412-3438.

Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S. H., & Ratti, C. (2014). Quantifying the benefits of vehicle pooling with shareability networks. Proceedings of the National Academy of Sciences, 111(37), 13290-13294.

Sak, H., Senior, A. and Beaufays, F., 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Fifteenth annual conference of the international speech communication association.

Schrank D, Eisele B, Lomax T (2012) Texas Transportation Institute 2012 Urban Mobility Report (Texas Transportation Institute, A&M University, College Station, TX).

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

Spieser, K., Samaranayake, S., Gruel, W. and Frazzoli, E., 2016, January. 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).

Stiglic, M., Agatz, N., Savelsbergh, M. and Gradisar, M., 2018. Enhancing urban mobility: Integrating ride-sharing and public transit. Computers & Operations Research, 90, pp.12-21.

Tao, C.C., 2007, September. Dynamic taxi-sharing service using intelligent transportation system technologies. In Wireless Communications, Networking and

Mobile Computing, 2007. WiCom 2007. International Conference on (pp. 3209-3212). IEEE.

Vakayil, A., Gruel, W. and Samaranayake, S., 2017. Integrating Shared-Vehicle Mobility-on-Demand Systems with Public Transit (No. 17-05439).
Van Rossum, G. (2014). Python 3.4. 3.

Wei, D., Chen, F. and Sun, X., 2010, June. An improved road network partition algorithm for parallel microscopic traffic simulation. In Mechanic Automation and Control Engineering (MACE), 2010 International Conference on (pp. 2777-2782). IEEE.

Winter, S. and Nittel, S., 2006. Ad hoc shared-ride trip planning by mobile geosensor networks. International Journal of Geographical Information Science, 20(8), pp.899-916.

Yan, S., Chen, C.Y. and Lin, Y.F., 2011. A model with a heuristic algorithm for solving the long-term many-to-many car-pooling problem. IEEE Transactions on Intelligent Transportation Systems, 12(4), pp.1362-1373.

Zhang, R. and Pavone, M., 2016. Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. The International Journal of Robotics Research, 35(1-3), pp.186-203.

Zhou, K., Fu, C. and Yang, S., 2014. Fuzziness parameter selection in fuzzy c-means: the perspective of cluster validation. Science China Information Sciences, 57(11), pp.1-8.