ABSTRACT

SYSTEMS FOR FREE PARKING ASSIGNMENT

by
Abeer M. Hakeem

Finding a free, curbside parking spaces in metropolitan areas, especially during rush hours, is difficult for drivers. The difficulty arises from not knowing where the available spaces may be at that time; and, even if the spaces are known, many vehicles may pursue the same spaces, causing serious parking contention and traffic congestion. This dissertation presents three cost-effective and easily deployable free parking assignment systems that optimize the travel time of the drivers.

The first contribution is the Free Parking System (FPS), a centralized solution that solves the curbside parking problem. Unlike existing solutions, FPS is cost-effective, as it does not need any sensing infrastructure. It relies on drivers' cooperation to maintain the parking availability information. FPS reduces parking space contention because it provides individual space assignments to drivers. The system consists of two components: a mobile app running on the drivers' smart phones that submits parking requests and guides drivers to their parking spaces, and a central server that manages the parking assignment process. The main novelty of FPS consists of its parking assignment algorithm, FPA, which combines a system-wide objective ("social welfare") with a modified compound laxity algorithm to minimize the total travel time for all drivers. The simulation results demonstrate that compared to a baseline solution, which mimics the way people search for parking today, and a greedy parking assignment algorithm, FPA reduces the total travel time for all drivers. Furthermore, FPA provides substantial improvements even when many parking spaces are occupied by drivers who do not use FPS.

The second contribution is the Distributed Free Parking System (DFPS), which solves the two intrinsic problems of the centralized FPS: scalability, as the server has

to perform intensive computation and communication with the drivers; and privacy, as the drivers have to disclose their destinations to the server. DFPS solves the scalability problem by using the smart phones of the drivers to cooperatively compute and forward to drivers the parking assignments, and a centralized dispatcher to receive and distribute parking requests. The parked drivers in DFPS are structured in a K-D tree, which is used to serve new parking requests in a distributed fashion. DFPS removes the computation and substantially reduces the communication handled by the dispatcher. DFPS solves the privacy problem through an entropy-based cloaking technique that runs on drivers' smart phones and conceals drivers' destinations from the dispatcher. DFPS provides a distributed version of FPA, which optimizes the total travel time for all drivers, while preserving driver's destination privacy. The evaluation demonstrates that DFPS obtains better travel time performance than a centralized system, while protecting the privacy of drivers' destinations and removing the computation and communication bottleneck from the server.

The third contribution is the Multi-Destination Vehicular Route Planning (MDVRP) system, which applies FPS to the multi-destination route planning problem. Specifically, MDVRP proposes an efficient solution for people in a city who drive their cars to visit several destinations, where they need to park for a while, but do not care about the visiting order. This instance of the multi-destination route planning problem is novel in terms of its constraints: the real-time traffic conditions and the real-time free parking conditions in the city. MDVRP uses TDTSP-FPA, a novel algorithm that finds the most efficient order to visit the destinations and also assigns free curbside parking spaces that minimize the total travel time for drivers. To evaluate MDVRP, a novel experimental platform that simulates real, multi-destination driver trips of over two million drivers, is built. Experimental results from a prototype implementation show that TDTSP-FPA delivers the best performance when compared to three baseline algorithms.

SYSTEMS FOR FREE PARKING ASSIGNMENT

by
Abeer M. Hakeem

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

August 2020

APPROVAL PAGE

SYSTEMS FOR FREE PARKING ASSIGNMENT

Abeer M. Hakeem

Cristian Borcea, PhD, Dissertation Advisor — Date
Professor, Computer Science, New Jersey Institute of Technology

---

Reza Curtmola, PhD, Committee Member — Date
Professor, Computer Science, New Jersey Institute of Technology

---

Xiaoning Ding, PhD, Committee Member — Date
Associate Professor, Computer Science, New Jersey Institute of Technology

---

Narain Gehani, PhD, Committee Member — Date
Professor Emeritus, Computer Science, New Jersey Institute of Technology

---

Tamer Nadeem, PhD, Committee Member — Date
Associate Professor, Computer Science, Virginia Commonwealth University

BIOGRAPHICAL SKETCH

Author: Abeer M. Hakeem

Degree: Doctor of Philosophy

Date: August 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2020
- Master of Science in Computer Science,
  King Abdul-Aziz University, Jeddah, Saudi Arabia, 2009
- Bachelor of Science in Computer Science,
  King Abdul-Aziz University, Jeddah, Saudi Arabia, 1997

Major: Computer Science

Presentations and Publications:

A. Hakeem, X. Ding, R. Curtmola, and C. Borcea, "Distributed Privacy Preserving Free Parking Assignment System", *IEEE Transactions on Mobile Computing (Under submission 2020)*

A. Hakeem, N. Gehani, X. Ding, R. Curtmola, and C. Borcea, "Multi-Destination Vehicular Rout Planning with Parking and Traffic Constraints", *in 16th European Alliance for Innovation International Conference on Mobile and Ubiquitous Systems: Computing and Networking Services (MobiQuitous'19)*, November 2019

A. Hakeem, N. Gehani, X. Ding, R. Curtmola, and C. Borcea, "Cooperative System for Free Parking Assignment", *In IEEE Vehicular Networking Conference (VNC)*, November 2017

A. Hakeem, N. Gehani, X. Ding, R. Curtmola, and C. Borcea, "On-the-Fly Curbside Parking Assignment", *in 8th European Alliance for Innovation International Conference on Mobile Computing, Applications and Services (MobiCASE'16)*, December 2016

إلى روح أبي الغالي.... رحمة الله عليه
إلى من قدَّمت سعادتي وراحتي على سعادتها... أمي الفاضلة
إلى أختي مها.... من علَّمتني أن الحياة من دون ترابط وحب وتعاون لا تساوي شيئًا
إلى من أتشوَّق لأن أرى مستقبلهم المشرق بإذن الله.. اولادي روحي وقُرَّة عيني ونبض فؤادي
إلى إخوتي.. وكل من يحبني بصدق وإخلاص
أهديكم خلاصة جُهدي العلمي

I dedicate my dissertation first to my father (Mohammed), who has taught me all of the important life lessons, who has stood by me through thick and thin. My father whom I look up to as my hero and has guided me through my life. My first teacher. I wish he was still here to support me and see me presenting this dissertation, but I know he will be looking down on me as I walk to accept my graduation diploma.

Secondly, I dedicate this to the woman who has put me in front of herself endlessly, loves me unconditionally, and shows her pride and encouragement to me day in and day out. This is for my astonishing, lovely mother (Samiha). I want to thank you for your endless support throughout this journey and want you to know I could not have made it without you. Now,
I want to thank my sisters (Hanan, Maha, and Eman) and my brothers (Hisham, Essam, and Samir Nawar). They have nursed me with advice, unselfish love, affection, and endless support. They have been by my side, reminding me to keep going.

They are my best friends and have shielded me with warmth and protection. I truly look up to them as my role models, and hope I can be even a fraction of how remarkable they are. Thirdly, I want to dedicate my dissertation to my kids (Lamar, Abdulrahman, Rama, Juwan, and Bateel). They inspire me to continue my work as mother and a PhD student. They are truly my treasures from my God. I do this for them. I give my deepest expression of appreciation for the encouragement that you gave and the sacrifices you made during my PhD. journey.

ACKNOWLEDGMENT

Completing a PhD is truly a long, difficult journey which would not possibly end successfully without the help and support of many people to whom I would like to express my gratitude. First and foremost, my advisor, Dr. Cristian Borcea, to whom I owe so much for this achievement. Dr. Borcea has been ceaselessly advising and guiding me since day one in both the research and writing phases of this journey. During the writing phase, he has been immensely patient, supportive, and motivational, enhancing every detail of the dissertation in order to perfect it. Despite my hardships, he has inexhaustibly stood by me and motivated me to persevere. I will forever remember and appreciate his constant support, encouragement, dedication, and sincere advice.

Next, I would like to thank all my PhD committee members from NJIT, Dr. Xiaoning Ding, Dr. Reza Curtmola, Dr. Narain Gehani for their valuable feedback and suggestions that vastly helped in improving the dissertation and papers. They continuously collaborated with me to form new, better ideas and advance the writing. They dedicated long hours to my research. I cannot show enough appreciation and gratitude for their dedication, patience, and time spent revising the papers.

I would like to thank Dr. Tamer Nadeem from Virginia Commonwealth University for being a part of my dissertation committee. His beneficial comments and suggestions have helped me improve the dissertation greatly.

More specifically, I would like to thank my fellow lab-mates Nafize R. Paiker, Mohammad Ashraf Khan, Hillol Debnath, and Nora Almalki, for their meaningful support during my research. Their valuable suggestions have continuously helped me clarify any confusion I experienced and have helped me improve my ideas.

Most importantly, I would not have achieved a fraction of this without the unconditional love and support of my family members. Although they are overseas,

they have striven to show me their constant support and have helped me tremendously when I was in need. And to my wonderful kids, whom I don't always have time to be with, have continuously stood by me. My family has supported my entire journey, and endlessly helped me, and for that, I am forever thankful.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

With the fast-growing number of vehicles over the past few years, especially in developing countries, finding a free curbside parking space has become a major problem for drivers in big cities. According to the geography of transport systems [1], searching for parking spaces can account for more than 10% of the local traffic in central areas of big cities, which leads to an average of 30% increase in traffic congestion. Meanwhile, in a 15-block survey area in New York, drivers cruised a total of 945,000 extra miles per year as they searched for curbside parking [2]. This accounted for a waste of over 47,000 gallons of gasoline and produced around 728 tons of carbon dioxide. The situation is worsening in developing countries where the number of vehicles has been increasing without sufficient investment in parking facilities.

Lacking enough visibility to determine where parking spaces are available exacerbates the parking problem. Consider the scenario where a person drives into a city center/downtown for an important appointment. The normal psychology of the driver is to reach the destination and park on a street that is close to the destination. Unfortunately, most people do not have sufficient time at their disposal to drive around cruising for available parking space after reaching the destination. As another scenario, the driver is going to a concert attended by many people. Naturally, she wants to find parking as close to the concert hall as possible. However, as she approaches the concert hall, the driver wonders if she should park as soon as she sees an empty space or if she should try to look for a closer space and potentially lose the empty space she previously spotted.

In the previous two examples, the drivers just want to visit one destination, and thus they need one parking space. There are, however, situations when people

have to go to several destinations in the same trip and need to park around each destination for a period of time. They do not care about the visiting order but may want to reduce their trip cost such as travel time consumption.

The general conclusion from all these examples is that effective parking management solutions need to be in place in order to help drivers find vacant parking spaces and avoid traffic congestion, air pollution, and waste of time. These solutions should aim to optimize the travel time for all drivers in the cities.

The rest of this chapter presents an overview of the free parking problem and our solutions in Section 1.1, and discusses the multi-destination route planning problem and our solution, when including parking and traffic constraints, in Section 1.2. The contributions of this dissertation are presented in Section 1.3. Finally, Section 1.4 details the structure of this dissertation.

1.1 Free Parking Assignment

1.1.1 Centralized Free Parking Assignment

With the advent of location-based services and embedded wireless sensors, applications that enable mobile devices to find vacant parking spaces in urban environments are being developed. A prime example of this type of application is SFPark [3]. It relies on 8,000 sensors embedded in the streets of the city of San Francisco, which can tell whether a parking space is available or not. The application shows a map with the available parking spaces in the driver's search area. The sensors cover about 25% of the available curbside parking in the city and cost USD $23M.

The primary goal of these applications is to help the individual drivers find open parking spaces, yet they have several shortcomings. First, deploying and maintaining the sensor infrastructure are costly. Second, drivers may not actually find vacant parking spaces by merely following the guidance. For instance, when the number of vacant spaces in an area is limited, many drivers, who obtain the same parking

information from the application, will head for the same set of spaces and only a few will park successfully. This will lead to congestion and driver frustration because the application does not attempt to provide individual guidance for drivers to specific parking spaces in order to minimize parking space contention. Third, parking space utilization becomes imbalanced: parking spaces for which information is provided are highly utilized and cause higher traffic congestion nearby, while other parking spaces may be routinely left vacant. In general, this type of application does not solve the basic parking problem. It would be better if the application just guided the drivers to exact locations where they are most likely to find open parking spaces. Then the following question arises: which algorithm should the application use to assign parking spaces to drivers?

To answer this question, the dissertation first studies the parking problem in a centralized model in which a sever makes parking choices for drivers and assigns each driver to a specific parking space with respect to her destination and a system-wide optimization objective, which aims to minimize the overall travel time for all drivers. To achieve the design goals, the Free Parking Assignment system (FPS) is proposed. FPS is cost-effective and does not need a sensing infrastructure, as it relies on drivers to cooperatively maintain parking availability information. FPS operates on-the-fly, as it handles new parking requests received over time and parking spaces that are found to be occupied by drivers who do not use FPS. It is important to note that FPS does not assume that all drivers use our system. It discovers the spaces occupied by unsubscribed drivers when the subscribed drivers report them. Then, it considers these spaces available after a time period based on the age of the observation reports.

1.1.2 Distributed Free Parking Assignment

Although the proposed centralized parking assignment solution solves the parking problem, the system suffers from two main limitations. First, the centralized server

requires substantial computation (to manage and assign free parking spaces) and communication with the vehicles (to receive requests and send responses) in real-time. This can be a bottleneck and for urban areas with many parking requests. Second, the parking assignment procedure is under risk of privacy violations. Specifically, the system requires drivers to disclose their destinations to the central server, so that parking spaces close to the destinations can be allocated to them. Such information may be utilized to infer private life details, such as the type of visited places or the number of visits at different places.

Conventional solutions, such as parking assignment games, where vehicles acting as players choose parking spaces in a competitive parking setting [4] are scalable. However, they assume that each vehicle has access to the location of other vehicles, which raises privacy concerns and has technical difficulties in real-time. The proposed parking assignment solution in [5] capitalizes on the ability of a trustworthy central controller to construct a feasible assignment in a distributed fashion via the coordination of drivers. The car-parking mechanism in this work is privacy-preserving in the sense that any car involved with the algorithm will not be able to find out the destination of any other car during the algorithm iteration. The problem with this solution is that the assignment computation and communication are burdens on the central controller. In addition, all exchanged information are stored in the controller which puts the private information at risk. Several techniques have been proposed for driver's privacy preservation, such as location perturbation and obfuscation [6], dummy location [7], and spatial cloaking [8, 9]. However, with these techniques, the assignment may not be optimal (*i.e.,* parking spaces may not be close to the destinations).

Therefore, this dissertation explores an efficient approach for designing a free parking assignment solution in a distributed mobile system, called a Distributed Free Parking Assignment (DFPS). DFPS aims to achieve driver's destination privacy with

low communication and computation overheads, and optimizes the total travel time of the drivers. DFPS solves the scalability problem by using the smart phones of the drivers to cooperatively compute and forward to drivers the parking assignments, and a centralized dispatcher to receive and distribute parking requests. Furthermore, DFPS solves the privacy problem through an entropy-based cloaking technique that runs on drivers' smart phones and conceals drivers' destinations from the dispatcher.

1.2 Multi-destination Route Planning

The aim of multi-destination route planning is to find the most efficient order of visiting a number of destinations in order to reduce the trip cost, such as the travel time. This problem has been studied extensively in the context of the Traveling Salesman Problem (TSP) [10, 11]. TSP is one of the most famous problems with route planning for multiple destinations. The goal of TSP is to find the shortest route that visits each destination once and returns to the original location. Although the TSP solutions can find a short path to multiple destinations, the concept of traffic constraint is not considered.

The Time-Dependent Traveling Salesman Problem (TDTSP) is a variation of TSP in which the amount of time it takes the salesman to travel from one destination to another fluctuates depending on the time of the day. By allowing the travel time between destinations to vary, TDTSP can better model real-world conditions such as heavy traffic, road repairs, and automobile accidents. We are interested in the time dependent problem introduced by [11–13], which strives to find the shortest route when the travel time depends on the time of day when the route is traversed. TDTSP is an efficient algorithm for routing problems, but to the best of our knowledge, none of the methods developed so far solve the multi-destination route planning problem with multiple real-time constraints such as parking and traffic. The problem is two-fold: a route planning problem and a free parking assignment problem.

Managing the interplay between traffic conditions and parking conditions to reduce the travel time for drivers can help both delivery companies and individuals in a city. For example, many times, delivery drivers must park around their destinations (*e.g.,* big buildings) where they need to deliver several packages. An individual, on the other hand, may have a number of tasks to do in a weekend day: grocery shopping, take clothes to/from dry cleaning, stop by the work office to get some papers, and see a small art exhibition downtown. The tasks can be done in any order, and we want to do it as efficiently as possible.

This dissertation presents a centralized solution that is able to efficiently plan routes for all drivers while satisfying the free curbside parking conditions (*i.e.,* provide parking guidance).

1.3 Contributions of Dissertation

This dissertation introduces three main contributions: two cost-effective and easily deployable free parking assignment systems (*i.e.,* centralized and distributed) that optimize the total travel time for all drivers, and a multi-destination vehicular route planning system that solves the multi-destination vehicular route planning problem, with parking and traffic constraints.

1.3.1 FPS: Free Parking System

We designed and developed a centralized free parking assignment system, FPS, that solves the shortcomings in the current parking guidance solutions. FPS has two components: a mobile app running on drivers' smart phones and a server, which is responsible for assigning parking spaces to drivers and providing individual parking guidance. In addition to submitting parking requests and providing parking guidance to drivers, the app reports to the server when a car is parked and when it leaves a parking space using input either from the drivers or from an activity recognition

algorithm based on phone sensors (*e.g.,* accelerometers and GPS). The server manages information about available parking spaces and handles parking requests in such a way as to optimize the social welfare system objective (*i.e.,* the total travel time for all drivers).

FPS employs a novel free parking assignment (FPA) algorithm to achieve this goal. FPA uses the social welfare criterion to solve driver contention for the same parking spaces in such a way as to minimize the total travel time to the destinations. FPA delays the parking space assignment as long as possible in order to accumulate more parking requests and thus perform a more efficient assignment. We created a modified version of the compound laxity algorithm [14] to determine how long a request can be delayed before it must be assigned a space. Our algorithm minimizes the total driving time to the parking spaces. By combining social welfare and compound laxity assignments, FPA is able to minimize the total travel time for all drivers.

FPS has been evaluated using two baseline assignment algorithms and two versions of FPA: (i) a naive algorithm that assumes a breadth-first-search for parking spaces around the destinations; (ii) a greedy algorithm that assigns the closest available space to the destination as soon as the driver enters a predetermined parking space allocation area; (iii) a basic FPA version that considers spaces occupied by unsubscribed drivers to remain occupied forever; (iv) an enhanced FPA version, FPA-1, that re-considers the spaces occupied by unsubscribed drivers after a time period. The results demonstrate that FPA reduces the total travel time by more than 4 times when compared to the naive algorithm and by 42% when compared with greedy algorithm, when all the drivers use our system. FPA also provides substantial improvements even when 25% of the spaces are occupied by unsubscribed drivers, and FPA-1 performs the best among all algorithms in this scenario. For example, FPA-1 reduces the travel time by 52% compared to greedy.

1.3.2 DFPS: Distributed Free Parking System

We designed and implemented DFPS, a distributed free parking system for assigning free curbside parking spaces to cruising drivers in cities. DFPS solves the two main problems with FPS: scalability due to its centralized architecture, and privacy due to the server knowing the destinations of all drivers. DFPS has three features: (1) a scalable system architecture for distributed parking assignment; (2) a distributed parking assignment algorithm among drivers that cooperate to efficiently assign parking spaces to drivers; and (3) a privacy-aware parking request generation that protects the privacy of drivers' destinations.

DFPS has two components: a mobile app running on drivers' smart phones and a dispatcher running at a server that enables cooperation among phones. The mobile apps on the phones form a distributed system that manages and assigns free curbside parking spaces. This substantially reduces the communication and computation overhead on the central server. The parked drivers in DFPS are structured in a K-D tree [15] based on their locations. This structure allows high efficiency in serving new parking requests through parallel processing. The K-D tree also provides for localized distributed computation and communication, which makes DFPS scalable.

To conceal drivers' destinations from the central dispatcher, DFPS uses a novel entropy-based spatial cloaking technique, where each driver can entertain parking assignment services without revealing her real destination and without seeking help from any centralized third party. In addition to spatial cloaking [8,9], techniques such as location perturbation and obfuscation [6] and dummy location [7] can also solve the problem of location privacy protection. However, they cannot be used in our settings because they may lead to parking assignments far from destinations [16,17]. The basic idea of our entropy-based cloaking technique is that each driver submits her parking request with a cloaked region as her destination, instead of her real destination. Specifically, for each real destination, the entropy-based cloaking technique selects the

nearest neighbouring destinations to construct a cloaked region, which contains both the real destination and the selected new destinations. The cloaked region must satisfy a $k-$anonymity privacy requirement: in addition to the real destination, the region must have at least another $k-1$ possible destinations that are not distinguishable from the real destination. When constructing a region, an entropy of distance method [18] is employed to avoid the clustering problem (*i.e.,* multiple destinations clustered in a small area, making it easier for an attacker to exploit the driver's destination). The method selects $k-1$ destinations that are evenly distributed to form the cloaked region. The technique also requires that the cloaked region contains at least a minimum number of available parking spaces to ensure that a parking space close to the real destination is likely to be available when the driver approaches it.

Similar to FPS, DFPS does not assume that all drivers use our system. It relies on subscribed drivers to submit observation reports regarding the parking spaces occupied by drivers that are not part of our system. DFPS avoids allocating the reported spaces for a period of time proportional with the age of the observation reports; then, it reconsiders these spaces.

DFPS is evaluated through multiple experiments. The results show that DFPS scales well. In particular, it eliminates all computation from the centralized dispatcher and reduces its communication load by a factor of two. In terms of average travel time, DFPS can decrease the average travel time by 26% compared to the centralized system, in addition to not disclosing the drivers' destinations to the dispatcher.

1.3.3 MDVRP: Multi-destination Vehicular Route Planning System

The last contribution of the dissertation defines a new instance of the multi-destination route planning problem, which has significant practical applicability. To the best of our knowledge, this is the first work on route planning that considers simultaneously the real-time conditions of vehicular traffic and free parking

availability. The main novelties of this work are: (1) the design and implementation of MDVRP system, and (2) TDTSP-FPA algorithm to manage the multi-destination route planning problem. The optimization goal of the algorithm is to minimize the total travel time for all drivers, where this time includes both the driving time to parking spaces and walking time between parking spaces and destinations. The design of MDVRP is modular and, thus, other algorithms for time-dependent route planning and parking assignment can be used to replace TDTSP-FPA; (3) we build a new experimental platform for realistic simulations of multi-destination routing. We use real vehicular mobility traces from over two million drivers from the city of Cologne, Germany to learn the spatio-temporal distribution of real driver destinations. Our platform then uses a new method to generate realistic multi-destination route requests, exploiting Cologne's road network along with many destinations and curbside parking spaces in the city's downtown.

The design of MDVRP has two components: a mobile app running on the drivers' smart phones and a server running in the cloud. The app submits real-time route requests to the server, receives optimized routes from the server, and guides the drivers toward destinations. In addition, the app reports to the server when and where a car is parked and when it leaves its parking space. This allows the server to manage the parking information and assign parking spaces to drivers. The server's main job is to interact with the mobile apps of all drivers and to optimize the routes for these drivers to reduce their travel time, while managing traffic congestion. The optimization determines the best order to visit the destinations and finds the best free curbside parking spaces for the drivers.

MDVRP uses TDTSP-FPA, a novel algorithm that combines a solution for the Time-Dependent Traveling Salesman Problem (TDTSP) [19] to find the fastest route for the next destination with our Free Parking Assignment Algorithm (FPA) to find free curbside parking that minimizes the driving plus walking time for all drivers in

the system. TDTSP-FPA manages the incoming requests in two steps: first, it finds the shortest path to the next destination in a trip in such a way as to minimize the total travel time. Second, it solves driver contention for the same parking spaces in such a way as to minimize the total travel time for all drivers. The travel time for one driver is the sum of: (1) driving time from the moment the driver submits a parking request to the moment she parks, and (2) walking time from the parking space to the destination and back. TDTSP-FPA's optimization goal is to reduce the total travel time for all drivers.

According to the experimental results, TDTSP-FPA reduces the total travel time by 34% when compared to the solution that represents current driver habits HTPO and by 29% and 26% when compared to baseline solutions for TSP and TDTSP, respectively. TDTSP-FPA scales well, as it works better when a larger fraction of drivers in the road network are MDVRP drivers. For example, TDTSP-FPA's travel time reduction compared with TDTSP's is 25% when 5% of drivers are part of MDVRP vs. 19% when only 3% of the drivers are part of MDVRP. The system is robust and provides benefits even when drivers do not comply with the recommended visiting order, but accept the parking assignment.

1.4 Structure of Dissertation

The subsequent chapters of this dissertation are structured as follows: Chapter 2 reviews related work. Chapter 3 presents FPS, a centralized parking assignment system. Chapter 4 describes DFPS, a mobile distributed system for free parking assignment. Chapter 5 describes MDVRP, a multi-destination vehicular route planning system. Finally, the dissertation concludes in Chapter 6.

CHAPTER 2

RELATED WORK

There are a large number of research work on different aspects of intelligent parking systems, which include occupancy detection, parking management, system development, dynamic pricing, etc. In this chapter, the reviewed papers are classified based on their specific approach, focusing on parking guidance information, parking assignment in both centralized and distributed architecture, and privacy preserving parking assignment. We also discuss the relevant work for the multi-destination vehicular route planning problem, with parking and traffic constraints.

2.1 Parking Guidance Information

In the last few years, extensive research efforts have been dedicated towards finding efficient means to aid drivers in their search for free curbside parking spaces, especially in highly solicited and crowded urban areas. To this end, one crucial piece of information required to decide which parking space to select is about parking occupancy or spaces' availability. With the advent of location-based services and embedded wireless sensors, several smart applications have been developed in order to assist drivers in their parking search. Proposed solutions in literature fall into two main categories: parking solutions with infrastructure assistance, and parking solutions relying on estimated/predicted information. In the first category, existing or added facilities gather accurate data about parking occupancy and capacities in order for drivers to efficiently locate their parking spaces. Whereas, in the second category, such privilege no longer exists and the status of the parking spaces is either predicted or estimated with other methods.

The relevant examples of solutions with infrastructure assistance are [3, 20, 21]. In SFpark [3] and SmartParking [20], where each parking space is equipped with

a fixed sensor to determine its occupancy/availability. The infrastructure then advertises the available parking spaces and manages their reservation. Moreover, a penalty mechanism is proposed in [20] to ensure that vehicles respect their assigned spaces. However, the deployment of these solutions is very expensive when monitoring the curbside parking spaces. For instance, in SFpark, the sensors that installed into the asphalt and cover about 25% of the available curbside parking in the city and cost $23M. When a user wants to find a parking space in some area of the city, the application shows a map with marked locations of the open parking spaces in the area. This necessitates a large installation and operational cost in order to adequately monitor the parking spaces at a city-wide level, or even at the level of a downtown area. ParkNet [21] proposed reducing the number of required infrastructure/sensors. Their idea was to provide a set of special vehicles (such as caps or buses) with ultrasonic sensors. These devices are used to determine and reserve vacant spaces even in isolated areas of the road. Although the authors show that these monitoring approaches are very effective and convenient, they have several shortcomings. First, the cost involved in deploying and maintaining the sensor infrastructure is high. Second, the precision of these ultrasonic devices lacks accuracy. Third, the concept of the solution itself implies that the designated vehicles continuously monitor the state of the road checking for parking availability. Fourth, all drivers see the same map at any given time, and many of them will compete for the same spaces. This will lead to congestion, drivers' frustrations, and parking contention problems. Finally, drivers have to shift their focus from the road to the map in their mobile devices to decide which space to choose from all available spaces. It would be more sufficient and safer if the app just guided the driver to an exact location where she will most likely find an open parking space.

As an example of a solution based on predictability, Verroios et al. [22] used vehicular ad-hoc networks (VANETs) as vehicles navigate through urban road

networks to search for open parking spaces. They presented an algorithm based on the time-varying Traveling Salesman Problem to compute a route of a driver that goes through all the parking spaces that are considered available. Their approach considered a probability of successful parking within a certain distance from the current location. However, this solution is difficult to apply in reality because the availability of the parking spaces can change at any time. In addition, even if a driver is successfully guided to a parking space, such a system in the aforementioned solution increases the probability of finding any parking space at the expense of missing the opportunity for a better space. Wolfson et al. [23] is another example that focuses on P2P dissemination of parking reports and presents a parking choice algorithm to choose parking spaces based on a relevance metric that includes the age of the open parking report. Their work assumes that a driver knows the expected time the slot will remain available from now, and how long it will take to travel there. In the solution presented by Bessghaier et al. [24], vehicles exchange information about both available and occupied parking spaces in cities. Based on the preferences of the driver, a decision module selects an appropriate parking space (in the experimental evaluation, parking spaces closer to the current location of the vehicle are preferred) and stops diffusing information about that parking space in order to maximize the chance of finding it open. Parking payment terminals (parking automates) are also used in [25] to disseminate information about available parking spaces. These types of solutions suffer from accuracy and scalability problems since this process needs to be iterated repeatedly for each freed parking space. It can also lead to the parking contention problem and traffic congestion.

The proposed solutions in this dissertation differ from the above research by three aspects. First, they do not rely on expensive infrastructure; instead rely on cooperative smart phones, which is a cheaper, more convenient, and more flexible alternative. They choose to learn the parking information from the drivers and from

cost-effective parking monitoring solutions. As an example, Nawaz et al. [26] proposed a smart phone based sensing system that leverages the ubiquity of WiFi beacons to monitor the availability of street parking spaces. Salpietro et al. [27] developed Park Here!, a smart curbside parking system based on smart phone-embedded sensors and short range communication technologies. Arnott and Rowse [28] developed an integrated model for curbside parking and traffic congestion control in a downtown area. Second, they aim to allocate parking spaces to reduce parking contentions: a scenario where multiple drivers are looking for a parking in a crowded area. Third, they guide drivers to their assigned spaces.

2.2 Parking Assignment

The next hurdle is how to efficiently assign parking spaces to drivers while benefiting from such information. For this purpose, several solutions have been proposed. We can classify these solutions into two main categories based on the decision of the maker's identity: centralized parking decision making (in which a central authority makes parking decisions and assigns each driver to a specific parking space), and distributed parking decision making approaches (where drivers are not passive and responsible for both searching and selecting vacant parking spaces).

2.2.1 Centralized Parking Assignment Model

In this model, drivers start by emitting their requests for parking spaces to a central parking authority manager. Different parking requirements can be specified by parking costs, proximity to the destinations, and any other parameter reflecting a specific driver's requirement. The central manager processes the received requests and contributes the assignment of the available open parking spaces to drivers. The parking assignment decision making module generally takes into consideration the drivers' requirements and the overall social welfare to maximize. For instance,

Mackowski et al. [29] developed a demand-based real-time pricing model to allocate parking spaces in busy urban centers optimally. Ayala et al. [30] developed a pricing model to minimize the system-wide driving distance. However, the proposed pricing approach is off-line in nature, as the number of vehicles and resources are known in advance and do not dynamically change. The reservation system for parking spaces is studied in [31]. A server collects information from curbside units and other vehicles, and reserves spaces for vehicles. This system attempts to circumvent the contention for parking spaces by using reservations; however, it does not optimize some system-wide objectives ("social welfare"). Basu et al. [32] presented a travel distance based approach, which is to assign the parking space to the nearest driver. However, this work assumes that the nearest driver will arrive earlier, which ignores the real-time traffic information.

In addition to the academic research, the parking assignment apps have addressed the parking problem by finding and reserving parking spaces to drivers. For example, SpotHero [33], allows a driver to book discounted parking in lots and garages right on her phone. Pango [34] offers the possibility of drivers paying for meter parking on their phone. ParkMe [35] and BestParking [36] are search engines that let drivers search for lot and garage availability and directs drivers to the best and cheapest parking options near to their destinations. These solutions are restricted, they solve the garage and meter parking problem only, and also do not consider the overall global social welfare. Unlike the previous works, this dissertation proposed a centralized Free Parking Assignment system FPS which does not require any pricing data as it deals with free spaces. FPS adapts on-the-fly to new parking requests and combines a system-wide social welfare objective with a modified compound laxity algorithm to minimize the total travel time for all drivers (walking and driving).

Even though the parking assignment problem has been solved in the centralized model, it may suffer from intrinsic problems. First, it is prone to an inherent single

point of failure problem. Second, performing intensive computation (to assign spaces to drivers) and communication with drivers (to receive location updates and guide them to exact spaces) in real time makes the centralised solutions infeasible for large regions with many drivers, which is not scalable. Third, drivers' sensitive information (*e.g.,* identities, destinations, etc.) have to be submitted for the the availability of parking spaces in their destinations, and this could result in privacy violation if they are not protected. In contrast to existing centralized solutions, different parking assignment solutions with advantages of decentralization, privacy, and trust has been utilized for different parking applications.

2.2.2 Distributed Parking Assignment Model

In the distributed parking assignment model, vehicles/drivers are not passive during the parking assignment process. Instead of relying on a central server's decision maker, they are responsible for both searching for open parking spaces and selecting the parking to which they prefer to access. Delot et al. [37] proposed a solution where each vehicle leaving its parking place becomes a coordinator for it. After collecting information among interested neighbors, it decides on which one to share the parking coordinates with. This process aims to reduce competition between vehicles in search for parking since only the elected vehicle knows the parking place's exact location. However, this solution suffers from scalability since this process needs to be iterated repeatedly for each open parking space. Moreover, it does not address how free parking spaces are being assigned in the initial process. Other parking assignment solutions such as the one proposed in [4] Ayala et al. propose a parking slot assignment game where vehicles acting as players choose parking slots in competitive parking settings. In [38], the same authors propose another parking space assignment approach denoted as GPA for Gravity-based Parking Algorithm. The basic idea behind the second approach is to use a heuristic based on the forces of

attraction exerted by the slots. Authors assumed that (1) each driver has access to the location of other vehicles, which raises privacy concerns and has technical difficulties to perform the real-time tracking; (2) drivers are distributed uniformly across spaces which does not happen in reality. They presented driving distance as a traveling cost while ignoring the real-time traffic information. The parking assignment system in [5] capitalizes the ability of a trustworthy central controller to construct a feasible assignment in a distributed fashion via the coordination of drivers. The problem with this solution is that the assignment computation and communication are burdens on the coordinator. This dissertation argues that the proposed Distributed Free Parking Assignment system DFPS provides scalability and adaptability. DFPS reduces the communication and computation cost by offloading the assignment process to the parked drivers, where each one manages and assigns drivers to spaces in their regions as well as it optimizes the system cost (*i.e.,* total travel time (walking and driving)).

2.3 Privacy-Preserving Parking Assignment

There is always a trade-off between privacy and disclosure of information. On the one side, the amount of the information gathered directly affects the effectiveness of the system. On the other hand, disclosure of information violates a driver's privacy (*e.g.,* real identity, destination, etc.). To support driver's privacy, a variety of privacy-preserving techniques have been proposed. These techniques are based on one of following concepts. (a) reporting false location (*i.e.,* dummy) [7] where the main idea is to report the fake location; (b) spatial cloaking [39] where the main idea is to blur a user's exact location into a cloaked region that satisfies certain privacy requirements, *e.g.,* $k-$anonymity (*i.e.,* the cloaked region contains k users) and minimum area A_{min} (*i.e.,* the cloaked region size is at least A_{min}). This technique is the most popular one and it supports many environmental settings, *e.g.*, centralized [40, 41], distributed [42,43], P2P [44], wireless sensor networks [45], and many problem settings

such as snapshot queries [40, 44, 46], continues queries [47], and trajectories [48]. Different from existing solutions, cloaking is utilized in an unique way in DFPS. First, existing solutions use it to break the linkability between users and their location and/or queries, while our solution aims to protect drivers' destinations. Second, existing solutions rely on a third party agent to perform cloaking in a centralized way or a peer-to-peer infrastructure for mobile users to perform cloaking collaboratively. DFPS does not assume these architectures, since it mainly uses the smart phone of each individual driver.

Anonymity algorithms are proposed to form spatial cloaked regions. Abul et al. [49] proposed a quad-tree-based anonymity algorithm which adopts a recursive method to continuously divide the space region in which the mobile user resides into four quadrants. Mokbel et al. [50] proposed an anonymous algorithm based on the Casper model, which effectively improves the performance of the anonymity algorithm in [39]. However, there are some problems with these algorithms. In [49], the anonymity algorithms may form redundant regions in the process of constructing an anonymous region. In [50], the distribution of users is not considered and due to the lack of users in sparsely populated regions, the anonymous region will fail to be constructed. Our proposed privacy technique works well for both sparse and dense regions. It considers the distance between the real destination and its neighbouring destinations to construct a cloaked region that satisfies $k-$anonymity and ensures that the destinations in the cloaked region are not clustered together.

Different works have been proposed for smart parking systems to preserve driver's privacy. For instance, the schemes [16, 17], proposed a centralized privacy-preserving parking reservation services. These schemes preserve the privacy of drivers' desired identities using anonymity. Also, they use location obfuscation techniques (*e.g.*, geo-indistinguishably and cloaking) to protect the drivers' real destinations. However, the location obfuscation techniques reduce the accuracy of

selecting nearest parking during the reservation process. Ni et al. [51] presented a smart parking navigation where users are guided by a cloud server and road side units (RSUs) to available parking lots in their destination. The scheme mainly preserves drivers' privacy by using anonymous credentials. However, hiding drivers' real identities is not enough because the cloud server can identify the drivers from their parking locations. Moreover, the drivers reveal sensitive information, such as current locations, destinations, and arrival times to the cloud server. This enables cloud servers to track drivers easily. Different from existing schemes, DFPS made a balance between driver's privacy and parking assignment effectiveness by leveraging pseudonymity to protect drivers' identities and entropy-based cloaking techniques to protect drivers' desired destinations. Existing solutions rely on a third party agent to perform cloaking in a centralized way or a peer-to-peer infrastructure for mobile users to perform cloaking collaboratively. DFPS is able to blur the drivers' destinations in cloaked regions without using any fixed communication infrastructure or centralized/distributed server. DFPS generates the cloaked regions to the smart phone of each individual driver.

2.4 Multi-destination Vehicular Route Planning

Vehicle route planning has been proposed as a strategy to decrease road traffic congestion and implicitly reduce the travel times for drivers. Most of the previous studies on route planning focused on single-destination scenarios. Unlike these studies, this dissertation focuses on a new and practical problem. Many drivers have to go to several destinations in a trip, but do not care about the visiting order of these destinations. Furthermore, our problem needs to satisfy real-time constraints regarding vehicular traffic and free curbside parking availability.

The Traveling Salesman Problem (TSP) is a well-known multi-destination route planning problem that aims to find the shortest route (*i.e.,* in terms of distance) that

visits each destination once [10]. Although this problem is NP-hard, there is a large number of algorithms that can solve the problem exactly for a practical number of destinations or approximately for a very large number of destinations. However, these algorithms assume that the travel times are constant throughout the day. The Time-Dependent Traveling Salesman Problem (TDTSP) is a variation of TSP in which the amount of time it takes the salesman to travel from one destination to another fluctuates depending on the time of the day. By allowing the travel time between destinations to vary, the TDTSP can better model real world conditions such as heavy traffic, road repairs, and automobile accidents. This dissertation is interested in the time dependent problem introduced by [11–13] which strives to find the shortest route when the travel time depends on the time of day and when the route is traversed.

In these real-world TDTSP problems, there are frequently additional constraints such as time-windows or precedence constraints. TDTSP with time windows [52] deals with finding a set of optimal routes for a fleet of vehicles in order to serve a set of customers, each one with a specified time window. Hurkala [53] proposes a novel algorithm that computes the minimum route duration for TDTSP with multiple time windows and time-dependent travel and service/visit time constraints. Different constraints are addressed in Huang et al. [54] to efficiently plan a route that satisfies deadlines and cost requirements. The work finds an objective-optimized route where the user-specified destinations are visited before their corresponding deadlines. It also considers multiple deadlines for multiple destinations as well as optimizing the trip cost simultaneously. Melagarejo et al. [55] proposes a set of benchmarks for TDTSP based on real traffic data and shows the importance of handling time dependency in the problem. The authors present a new global constraint (an extension of no-overlap) that integrates time-dependent transition times and shows that this new constraint outperforms the classical Constraint Programming approach. In addition to academic

research, route planning apps such as Route4Me, RouteXL, and GSMtasks [56–58] aim to optimize driver's route when traveling to multiple destinations. These apps are able to efficiently manage driver fleets as well as business and delivery drivers.

To the best of our knowledge, none of these works consider finding free curbside parking for drivers and does not consider the influence parking availability and parking locations on the traffic conditions. This dissertation addressed this issue by presenting Multi-destination Vehicular Route Planning system MDVRP. MDVRP system is the first work on multiple-destination route planning that considers real-time parking and traffic conditions for multiple destinations, while optimizing the total travel time for all drivers.

2.5 Summary

This chapter discussed the existing studies related to intelligent parking systems. First, we presented existing solutions for parking availability detection/prediction that rely on dedicated infrastructure and their shortcomings. Next, we have discussed existing work on curbside parking assignment in different models (centralized and distributed). We also discussed previous works related to driver's privacy. Finally, we presented related work to the multi-destination vehicular route planning problem.

CHAPTER 3

CENTRALIZED FREE PARKING ASSIGNMENT

This chapter provides a general overview of the basic design of the proposed free parking assignment system that provides individual parking space assignments to drivers in Section 3.1. Section 3.2 introduces the greedy assignment algorithm in order to emphasize the parking problems with this simple solution and motivate the need for a more complex assignment algorithm. Section 3.3 defines the assignment problem and the social welfare optimization criterion. The parking assignment algorithm is described in Section 3.4, and the evaluation results are presented in Section 3.5. The chapter is summarized in Section 3.6.

3.1 FPS Overview

To illustrate how the FPS works, let's consider the scenario from Figure 3.1, in which a driver requests free parking space next to her destination. The FPS system consists of two components, namely parking requester (PR) and parking allocator (PA). PR is a mobile app that runs on each driver's smart phone and is in charge of submitting parking requests, reporting parking status to PA, and guiding drivers to the assigned parking space. Each parking request contains the requesting driver's current location and the desired destination. The reporting of parking status relies on drivers manually registering their "parked" and "left parking space" status. Alternatively, the app can learn this status from both an activity recognition service running on the phone [59] and from a crowdsensing approach that utilizes the pedestrians' smartphones on-street to identify free curbside parking spaces [60]. The parking allocator (PA) runs on a central server, where it manages the incoming parking requests and aggregates the PR reports to determine the available parking spaces. For availability computation, PA assumes that not all drivers participate in our system, *e.g.,* not all drivers are

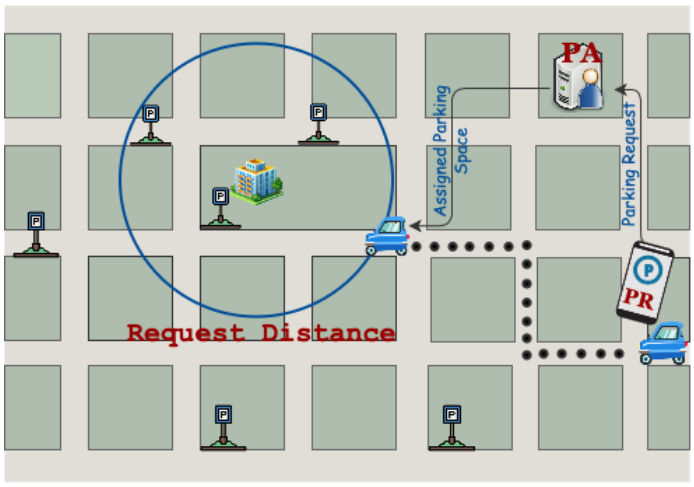

Figure 3.1 Parking request example.

equipped with the PR component. This means that some parking spaces are occupied by drivers that are not part of FPS. The FPA algorithms (see Section 3.4) running at PA discover and iteratively monitor these parking spaces. In addition, PA could estimate the number of spaces that are occupied by non-participating drivers in order to reduce the number of unsuccessful assignments [61, 62].

The basic idea of the FPS parking assignment is described as follows. Drivers who are looking for parking spaces use PR to send requests to PA. All incoming requests are streamed into a queue and are processed first-come-first-serve. For each request, PA allocates the available parking space that best matches the driver's destination. PA does not assign parking spaces to drivers who are far from the destination in order to reduce the likelihood of assigned parking spaces not being available upon the driver's arrival. Such a situation could happen due to unsubscribed drivers, and the likelihood that a space is taken by an unsubscribed driver increases over time. Therefore, FPS just informs the drivers that are far from their destinations that they will be assigned parking when they enter a zone of its destination, called *Request Distance* (see Figure 3.1). PR shows the drivers this area on the map, so they know when they should expect to receive a parking space as they approach the area. The *requests distance* is defined as a circle with the destination at its center. The size

of the request distance has to be large enough in order to avoid assigning a driver to a parking space outside the request distance or perform the assignment after the driver has passed the space. These two problems could increase driving time as well as reduce driver's satisfaction with the system. Therefore, we determined experimentally that the radius should be initially set to the average length of the roads within the whole region managed by a PA (*e.g.,* a zip code). Then, the radius is adjusted periodically based on the parking occupancy rate in the area: the radius is increased when the occupancy becomes higher. In our design, FPS sets the request distance on behalf of the drivers. However, the drivers could be allowed to set this distance themselves.

Once a driver enters the request distance, her parking request is scheduled for assignment and the assigned space is returned to the driver. FPS makes the assignment decision in such a way as to minimize the total travel time of the drivers.

3.2 Strawman Solution: Greedy

A strawman solution for the FPS's parking space assignment algorithm is a greedy strategy that minimizes the travel time for each individual driver on a first-come-first-serve basis. Unfortunately, this strategy cannot guarantee that the total travel time for all drivers is minimized. On the contrary, the greedy strategy may lead to substantial increases in the total travel time.

For example, consider the parking problem shown in Figure 3.2, in which edge labels represent travel time in minutes. The travel time for each driver is the sum of the driving time to the parking space and the walking time between the parking space and the actual destination. Greedy yields the (driver, parking space) assignment $(Driver1, space1)$, $(Driver2, space2)$, and a total travel time of 50 minutes. On the other hand, there is another possible assignment $(Driver1, space2)$, $(Driver2, space1)$ with a total travel time of 40 minutes. This requires $Driver1$ to drive to a farther space, $space2$, rather than driving to $space1$ which is closer. An assignment that

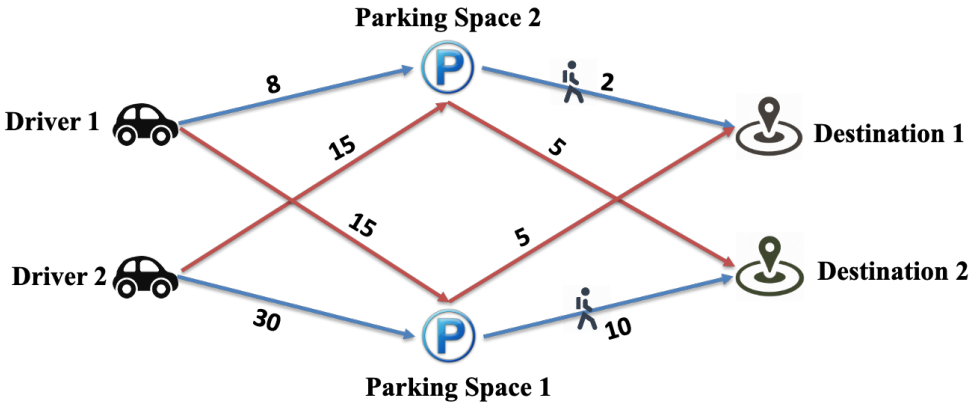

Figure 3.2 An example of parking assignment.

minimizes the overall total driving time is possible when a central authority can choose this parking assignment. We believe it is worth designing more advanced assignment algorithms that maximize the social welfare (*e.g.,* minimize the total travel time over all drivers) because they will lead to less pollution, less wasted time in congestion, and overall better travel time for all the drivers. This, of course, is achieved at the expense of slightly larger travel times for some drivers when compared to the greedy strategy.

3.3 Parking Assignment Problem Formulation

In this system, we consider a parking assignment problem defined as follows. Given a set of drivers, each of whom needs to reach a specific destination, and a set of curbside parking spaces, we would like to assign the parking spaces to drivers in order to satisfy a system-wide objective. Let $S = \{s_1, s_2, ..., s_m\}$ be the fixed set of curbside parking spaces distributed across a city region. Let $V = \{v_1, v_2, ..., v_n\}$ be the finite set of drivers that are trying to reach destinations in the considered city region. We assume the number of drivers is less or equal to the number of parking spaces. The drivers look for parking spaces close to their destinations, which include places such as banks, shops, houses, parks, hotels, and restaurants among others. Similar to the parking spaces, the destinations are geographically dispersed across

a city region. The drivers are assumed to be moving independently based on legal speeds and the congestion levels on different road segments. We also assume that each driver v_i's smart phone can compute the approximate driving time to her destination, $T_d(O_{v_i}, d_{v_i})$, simply based on the geographical distance between her original location O_{v_i} (*i.e.,* the location from where the parking request has been submitted) and the destination d_{v_i}. This information is attached to the parking request and is updated by driver's GPS as the driver approaches the destination.

The travel time for a driver v_i to reach her destination d_{v_i} includes two components:

- T_d $(O_{v_i},\ s_j)$ is the driving time of driver v_i from the moment she submits her request from location O_{v_i} until she parks at the parking space s_j.

- T_w $(s_j,\ d_{v_i})$ is the walking time of the driver from the moment she parks until the moment she arrives at her destination d_{v_i}.

Our goal is to determine an assignment Y of drivers to parking spaces that maximize the total system cost. The system cost is maximized by an assignment that minimize the following objective:

$$TC = \sum_{v \in V} TC(v) \tag{3.1}$$

TC indicates to a cost of the total travel time of all drivers to reach their exact destinations d_v. The computation of TC includes two phases:

Phase one: minimize the total walking time from the parking spaces to the drivers' destinations by using a strongly well-known minimum-cost network flow algorithm (*e.g.,* see [63]): *Find assignment* (v_i, s_j) *for all* $v_i \in V$, *s.t.* $min \sum_{i=1}^{n} T_w(s_j, d_{v_i}), where\ n = |V|, s_j \in S$.

Phase two: minimize the total driving time from the drivers' current positions to the parking spaces by using our modified compound laxity algorithm: *Assign* v_i *next if the laxity value* L_{v_i} *is min for all* $v \in V$.

In Y, the assignment of a driver v_i to a parking space s_j can be represented with a binary decision variable $y_{ij} : v_i \rightarrow s_j$:

$$y_{ij} = \begin{cases} 1, & \text{if } v_i \text{ is assigned to } s_j \\ 0, & \text{otherwise} \end{cases} \qquad 1 \leq i \leq n, 1 \leq j \leq m \tag{3.2}$$

$$\sum_{i=1}^{n} y_{ij} \leq 1, \quad 1 \leq j \leq m \ (i.e., s_j \in S) \tag{3.3}$$

$$\sum_{j=1}^{m} y_{ij} = 1, \quad 1 \leq i \leq n \ (i.e., v_i \in V) \tag{3.4}$$

Constrains 3.3 and 3.4 ensure, respectively, that a driver receives at most one space and that a space is not assigned to more than one driver. The violation of either constraint leads to invalid assignments, which are either wasteful (*e.g.,* assigning multiple parking spaces to the same driver) or infeasible (*e.g.,* multiple drivers sharing the same parking space).

3.4 FPA Algorithms for Parking Space Assignment

This section presents two versions of FPA, a dynamic parking assignment algorithm used by the FPS system to manage driver requests over time subject to social welfare optimization. The algorithm handles a set of driver requests coming to the system

independently by assigning available parking spaces to the drivers to satisfy their requests.

By reducing the problem of finding available parking spaces to an instance of the minimum-cost network flow problem on a directed bipartite graph, a strongly polynomial time can be achieved [4]. Although this method results in a minimum walking time and shows good computational properties, it can hardly meet our system-wide objective described in Equation (3.1) for two reasons. First, this method is designed for offline settings where the number of parking spaces and drivers are known and cannot be customized to a real-life, dynamic situation. Second, it only minimizes the total walking time.

Therefore, we propose a different algorithm to construct the parking assignment process dynamically over time and to maximize the social welfare described in Equation (3.1). The algorithm addresses two challenges. One is the selection of parking spaces, *i.e.,* which parking space should be assigned to each driver to satisfy her request, and the other is when a parking space should be assigned to a driver. To address the first challenge, the algorithm tries to assign to each driver the parking space closest to her destination. Assigning parking spaces far away from the destinations increases driving distance and/or walking distance. To address the second challenge, the algorithm assigns a parking space to a driver when she approaches the destination and is about to look for a parking space. Assigning parking spaces too early reduces the utilization of parking spaces. Assigning parking spaces too late may results in increasing driving time and bad user experience.

Specifically, FPS periodically examines and updates the status of the drivers and their requests. The period can be determined as a function of the road network structure, parking spots distribution, and parking requests distribution. In our simulations, we experimentally determined that a period of two seconds, which

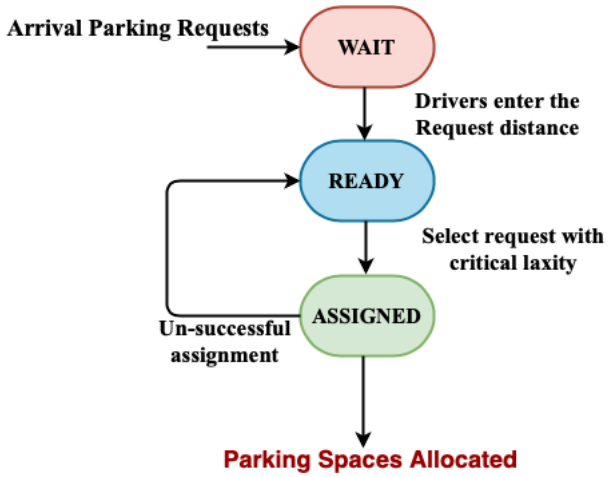

Figure 3.3 State transition of drivers in FPS.

provides a good trade-off between performance and overhead. FPS moves the requests through the states shown in Figure 3.3. These states are described below:

- WAIT: When a driver request comes to the system, it is stored in a FIFO queue, waiting to be scheduled.

- READY: when the driver moves into the request distance, the request is marked as *READY*. FPS schedules READY requests using FPA, which will be described later in this section.

- ASSIGNED: The request enters this state when it is selected by FPA and assigned to a parking space. The request stays in the *ASSIGNED* state until the driver successfully park in her assigned parking space. If the assigned parking space is found to be occupied by an unsubscribed driver when this driver tries to park there, the request moves back to the *READY* state with a high priority assignment. The request is finally removed from the system when the driver leaves the parking space. The app on the driver's phone will notify the system when the car leaves the parking space. To deal with the

case in which the notification is not received (*e.g.,* when the driver's phone is turned off or disconnected), each assignment has an expiration time, after which the request is also removed from the system and the parking space is deemed available again.

During each period, the main task of FPS is to select requests from the $READY$ state and assign them. This is the job of the FPA algorithm, and its main steps are described in Algorithm 1.

Algorithm 1 FPA Pseudo-code

1: Given a destination d_{v_i} and an estimated driving duration to destination D_i for each driver $v_i \in V$
2: **Preallocation:**
3: **for** each request v_i in $READY$ state **do**
4: Allocate to v_i the closest available parking space to d_{v_i}
5: **end for**
6: **Preallocation Adjustment:**
7: **for** each request v_i in $READY$ state **do**
8: **if** v_i shares a parking space with another request **then**
9: Find a new parking space for v_i that minimizes the total walking time
10: **end if**
11: **end for**
12: Update the *laxity* of each driver v_i in $READY$ state based on D_i and its currently allocated parking space
13: Search for a READY driver v with the minimum laxity value
14: **Assignment:**
15: Finalize the parking space assignment for v and change its state to $ASSIGNED$
16: Show the parking space on the smart phone of v.

For each request in the $READY$ state, FPA first pre-allocates to the driver the closest available parking space to her destination (lines 3-5). Then, it tests whether the pre-allocation can be a valid assignment for each request. The pre-allocation is valid if a parking space is not pre-allocated to more than one driver, as defined by both constraints 3.3 and 3.4. If it is valid, FPA continues with line 12. If not, the system immediately adjusts the pre-allocation by re-allocating other parking spaces to some of the drivers to remove the duplicated assignments of parking spaces (lines

7-11). We use the solution to the flow problem described in [4] to select parking spaces since it can minimize the total walking time.

Note that the pre-allocation and the adjustment of pre-allocation do not actually assign the parking spaces. The actual assignments are delayed and take place only when the requests become urgent (lines 12-13). The urgency is measured by the *laxity* value Lv_i of each request v_i, which is defined as follows:

$$L_{v_i}(t) = min(T_d(C_{v_i}, s_j), T_d(C_{v_i}, d_{v_i})) \tag{3.5}$$

where $T_d(C_{v_i}, s_j)$ is the estimated driving time of driver v_i from her current location C_{v_i} to the parking space s_j; and $T_d(C_{v_i}, d_{v_i})$ is the driving time of the driver v_i from her current location to her destination. The intuition is that a parking space must be assigned to a driver before she reaches either her destination or an available parking space close to her destination (represented by the parking space pre-allocated to her). Thus, the smaller the laxity value is, the more urgently the request assignment must be finalized. When the laxity values are calculated, we round the values to whole seconds. FPA compares the laxity values of READY requests and selects the requests with the smallest laxity value to finalize their assignments. The operations in lines 3-15 are repeated periodically to handle the remaining requests in the queues and the newly-arrived requests.

While we assume that FPS drivers are generally representative of the entire driving population, we do not assume that all or even a large fraction of drivers will use FPS. Also, we do not assume that many pedestrians walk on the street sidewalks to detect the parking availability using internal sensors (*i.e.,* magnetometer) of their smartphones [60]. Therefore, FPS drivers may compete for parking spaces with non-FPS drivers, which we call unsubscribed drivers. Figure 3.4 illustrates how

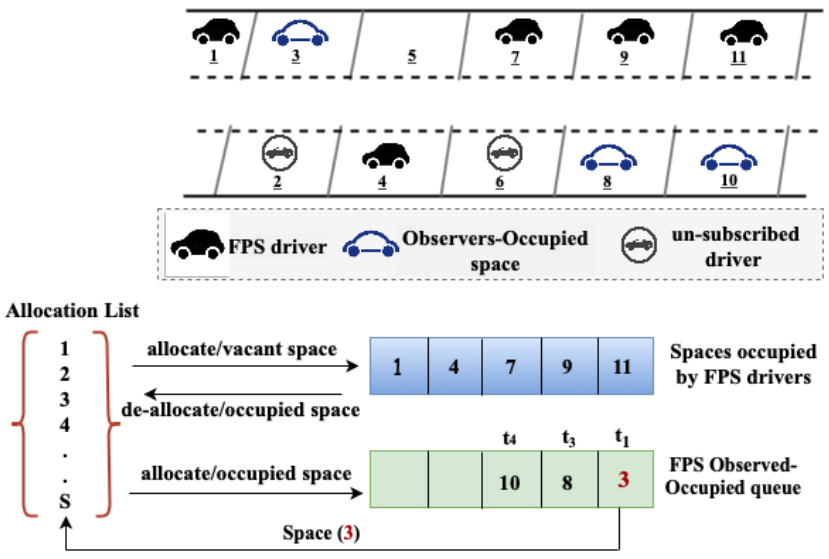

Figure 3.4 Illustration of parking spaces in different states.

FPS manages parking spaces. Since not all the parking spaces are available to FPS drivers, FPS needs to maintain a list of spaces that may be potentially available (spaces 2, 5, and 6 in Figure 3.4). Free spaces may be detected in different ways. For example, the mobile app of the subscribed drivers can inform FPS when they leave a parking space (*i.e.,* using algorithms based on analysis of GPS and accelerometer readings). Many vehicles are equipped with cameras, through which the availability of nearby parking spaces can be visually confirmed.

FPS also needs to keep track of occupied spaces and avoid assigning these spaces. While the spaces allocated by FPS itself can easily be maintained, there are spaces taken silently by unsubscribed drivers. For example, spaces 2, 3, 6, 8 and 10 are occupied by unsubscribed drivers in Figure 3.4. FPS relies on subscribed drivers to report these spaces to the system when they find that their parking spaces have already been occupied. When it receives such reports, FPS marks the spaces as "observed_occupied"; spaces 3, 8, and 10 are such examples in Figure 3.4. Then, FPS puts the requests of the drivers who reported these spaces back in the *READY* state.

In our example, FPS does not yet know that spaces 2 and 6 are occupied because no subscribed driver has reported them. Such spaces are called "hidden spaces".

Parking spaces marked as "observed_occupied" will not be assigned to other requests to avoid unsuccessful assignments. However, permanently marking parking spaces as "observed_occupied" inevitably reduces the utilization of parking spaces since "observed_occupied" spaces may become available later. To solve this problem, we propose FPA-1, an enhanced version of FPA to reclaim "observed_occupied" spaces. FPA-1 keeps track of how much time subscribed drivers occupy their parking spaces and maintains an average parking time value. Instead of this global average parking time, FPS could maintain per-street averages for higher accuracy. FPS assumes that "observed_occupied" spaces may also be occupied for similar amounts of time with the average parking time of subscribed drivers. When a space is reported to be taken by an unsubscribed driver, FPA-1 moves the space to a queue, named *observed_occupied queue*, and assigns a timer to this space, which expires after the average parking time. When the timer expires, the space is moved back to the allocation list (*e.g.,* space 3 in Figure 3.4).

3.5 Evaluation

This section evaluates the performance of FPA and FPA-1 when compared to Greedy and a Naive solution. Greedy assigns parking spaces to drivers as soon as they reach the initial parking allocation area in a first-come-first-serve manner. When selecting an available parking space for a driver, it always chooses the space closest to the driver's destination. The Naive strategy assumes the driver goes to the destination and, once there, she starts a breadth-first-search for parking spaces along the nearby road segments.

The evaluation is done via simulations over a real road network. The experiments simulate two different scenarios: *subscribed-driver-only scenario*, which

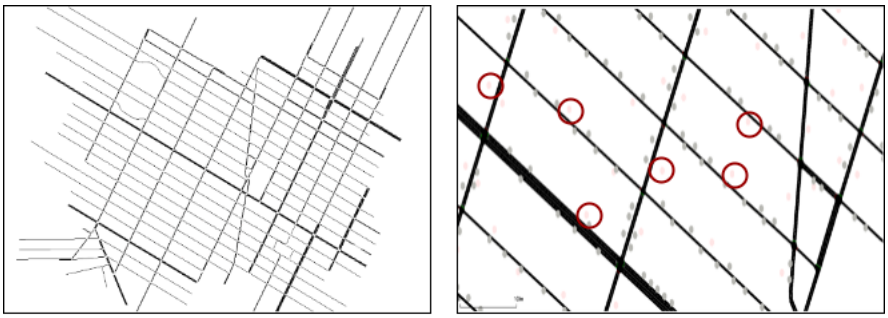

Figure 3.5 Road network used in experiments (a). Example of zoomed-in road segments (b): parking spaces (gray dots) and destinations (red circles).

assumes that all drivers in the system use FPS; *unsubscribed-driver-interference scenario*, which assumes there are a number of drivers who have not subscribed to FPS. In the second scenario, unsubscribed drivers may occupy, without notification, parking spaces known to the system as available.

We use *average travel time* metric to compare the performance of different assignment strategies. For each driver, it includes the time spent on driving to the parking space and walking from the parking space to the destination.

3.5.1 Simulation Setup

In our experiments, we use SUMO/TraaS [64], to simulate vehicles going to their destinations in a business district in Manhattan, New York City. The road network and the locations of curbside parking places are imported into the simulator based on the real map of the district. Figure 3.5(a) shows the road network used in the simulations, while Figure 3.5(a) illustrates an example of destinations and parking spaces along a few road segments. The total number of parking spaces is 1024, and the total number of destinations is 400.

The starting locations and the destinations of the vehicles are randomly chosen. However, the destinations are chosen from a small region in the center of the map to ensure enough contention for parking spaces. Each vehicle moves along its route at

the legal speed limit of each road on the route and the movement is restricted within the map. Every vehicle may adjust its speed for safety driving and to follow traffic laws. For example, it must keep a reasonable distance from the vehicle in front of it or it slows down when approaching an intersection or its parking space. Once a vehicle parks, we calculate the driving time and the walking time; For walking time, we consider an average speed of 1.4 m/s, which is reasonable for adults (men and women) [65, 66].

To simulate the scenarios with different parking densities and contention levels, we varied the number of vehicles, the number of parking spaces, and the number of destinations, which are as specified in each individual experiment. FPS starts each test with 1024 vacant parking spaces. The arrival rate of the requests falls within the range of 1 to 5 requests per second. The period for the parking assignment algorithm is set to 2s; this value was determined experimentally to provide a good trade-off between performance and overhead. For each experiment, we collected results from 5 runs and averaged them.

3.5.2 Results for Subscribed-Drivers-Only Scenario

Figure 3.6 compares the performance of FPA, Greedy, and the Naive algorithm by varying the number of drivers from 128 to 768 with a fixed number of destinations (8) distributed in the centroid area of the map.

The results demonstrate that FPA outperforms the comparison algorithms. When the number of drivers increases, the average travel time grows quickly for the Naive algorithm. This is because the contention for the parking spaces close to the destinations leads to substantial traffic congestion, which is exactly what we observe in real life. FPA decreases the average travel time by a factor of 4 compared with the Naive solution for 768 drivers (110.49 minutes). These results demonstrate the substantial impact FPS can have on driving and parking in the cities. As expected,

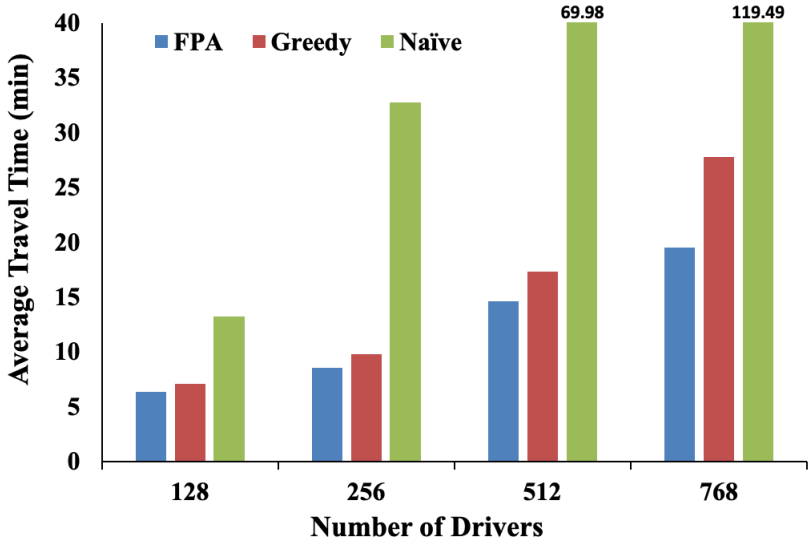

Figure 3.6 Average travel time for a different number of drivers and a fixed number of destinations (8).

the average travel time increases for FPA and Greedy with the number of drivers, but this increase is sub-linear. This is because these algorithms avoid having the drivers go to the destinations and then starting to search for parking.

Compared to Greedy, FPA is more effective as it reduces the average travel time by as much as 40%. These results can be explained by the design of FPA, which optimizes the system-wide travel time. As discussed, maximizing the social welfare leads to lower walking time, and our modified compound laxity algorithm leads to lower driving time.

Figure 3.7 shows the average travel time of 768 drivers when the number of destinations is varied from 1 to 8. The figure also plots the contribution of walking time and driving time in the total time. With more destinations, the advantage of FPA over Greedy becomes more prominent. Compared to Greedy, FPA reduces the average travel time by 18% in the one destination case and 42% in the 8 destination case. The reason is that, with more destinations, there is more space for FPA to perform optimization by balancing the driving and walking distances of the drivers

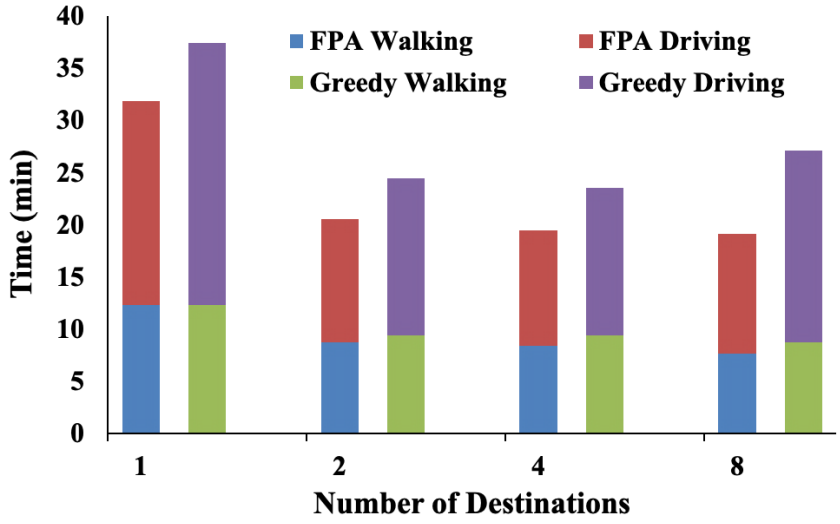

Figure 3.7 Walking and driving time of FPA and Greedy with a different number of destinations and a fixed number of drivers (768).

with different destinations. Thus, both average driving time and average walking time can be reduced with FPA. As shown in Figure 3.7, FPA can reduce the average driving time by up to 61% and reduce the average walking time by up to 14% relative to Greedy. We observe that Greedy with two and four destinations performs better than with eight destinations. The reason is that some parking spaces could be allocated for more than one destination and Greedy is not able to allocate them effectively (*i.e.,* similar to the example shown in Figure 3.2). This phenomenon becomes significant as the number of destinations increases to 8.

Since FPA minimizes the total travel time for all drivers, one may ask how is the performance of individual drivers impacted by our algorithm. To answer this question, we conduct an experiment to find out the travel time gains or losses for individual drivers. To measure the gains/losses, we calculate the ratio between the travel time obtained by the Naive algorithm and the travel time obtained by FPA for each driver. If the ratio is higher than 1, the driver has benefited from FPA. Otherwise, the driver has not. Then for each run of the experiment, we sort the

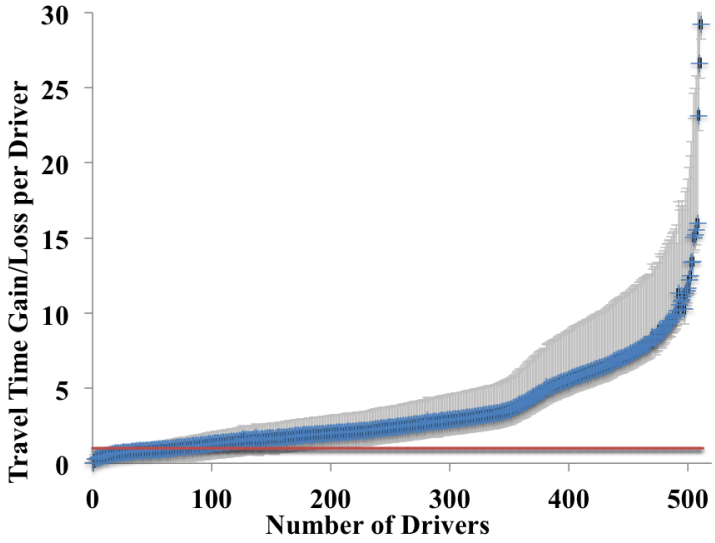

Figure 3.8 Distribution of travel time Gain/Loss for all drivers in the system with a fixed number of destinations (8). Gains are values greater than 1, and Losses are values less than 1. Error bars are shown.

drivers in the ascending order of these ratios. We then, average the ratios for these sorted drivers as shown in Equation (3.6).

$$\frac{\sum_{j=1}^{N}(T_i^j(Naive)/T_i^j(FPS))}{N}, \quad i \in V \tag{3.6}$$

where N is the number of runs, and T_i^j is the travel time for the driver in position i in the sorted driver list for experiment j for both Naive and FPS algorithms.

Figure 3.8 plots the distribution of individual travel time gains/losses for 512 drivers. The results show that 87.8% of the drivers obtain gains, and some of them have very large gains. Nevertheless, the number of drivers with losses is not negligible. From a practical point of view, a few bad experiences could impact the adoption rate of FPS. Therefore, we plan to investigate methods to limit the number of drivers who experience losses and bound the loss ratio to low values.

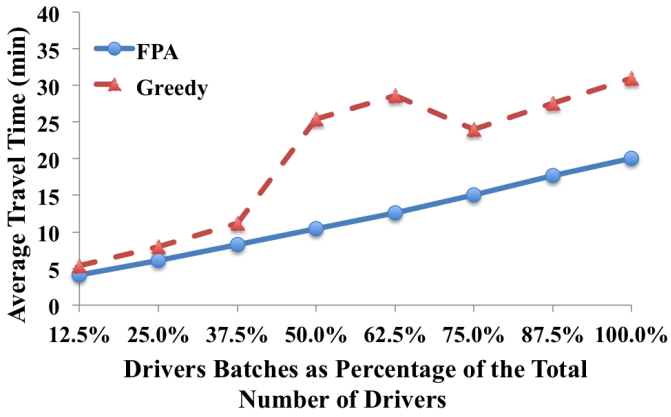

Figure 3.9 FPS consistency over time: 768 drivers divided into eight equal batches as a function of their arrival time at 8 destinations.

In the next experiment, we analyze the behavior of FPA and Greedy over time. We divide 768 drivers in 8 equal batches based on the time they arrive at their destinations.

Figure 3.9 shows that FPS performs consistently better than Greedy during the whole parking assignment process as new drivers enter the system over time. As expected, the average travel time for earlier batches is lower as there are more parking spaces available at locations closer to destinations when they arrive. Also, the difference between the two algorithms is not large because Greedy can perform a good assignment under these conditions. However, we notice that FPA performs substantially better than Greedy (up to 1.5 times) for the middle batches. Since Greedy simply moves each vehicle toward the closest parking space available, the total driving time and therefore congestion are higher, especially when the number of assigned vehicles increases and the number of available spaces decreases. Therefore, in Greedy, drivers waiting to be assigned are congested with drivers heading to their assigned spaces. The average travel time in the last three batches decreases because the assigned drivers park in their spaces and most of the vehicles on the road are waiting to be assigned. For the later batches, FPA is still clearly better, but it does

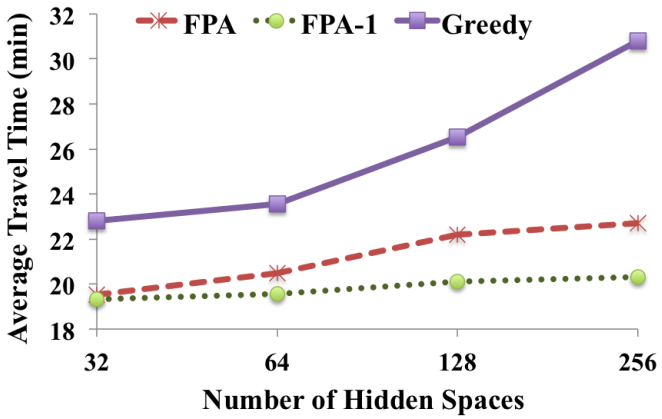

Figure 3.10 Average travel time when varying the number of hidden spaces: 768 drivers and a fixed number of destinations (8).

not have as much room for optimization as it has for the middle batches. This is because fewer parking spaces are available.

3.5.3 Results for Unsubscribed-Driver-Interference Scenario

To test the capability of FPS to tolerate interference from unsubscribed drivers, we randomly selected a number of spaces located in the request distance and marked them as "hidden" spaces, indicating that they are currently occupied by unsubscribed drivers (see Figure 3.4). FPS is not aware of a "hidden" space until a vehicle is assigned to that space and finds that the space is taken (*i.e.*, an "observed_occupied" space). During the experiment, the "observed_occupied" spaces become available over time to simulate unsubscribed drivers leaving their parking spaces. The times for the "hidden" spaces to become available are assumed independent and exponentially distributed, but the average parking time for unsubscribed drivers is same to the average parking time for subscribed drivers. As the number of "hidden" spaces increases, we increase the radius of the parking space allocation area proportionally, such that there are still enough parking spaces available to the subscribed drivers.

Figure 3.10 compares the performance of FPA, FPA-1, and Greedy when the number of hidden spaces is varied from 32 to 256. We observe that both FPA and FPA-1 outperform Greedy, and their relative performance when compared with Greedy increases with the number of hidden spaces. We also notice that FPA-1 achieves lower average travel time than FPA, and its performance is almost constant. FPA-1 reduces the average travel time by 10% relative to FPA and 33% relative to Greedy on average. These results demonstrate that FPA-1 adapts very well to the interference caused by unsubscribed drivers.

3.6 Summary

In this chapter, a centralized free parking assignment system FPS was presented. FPS is a cost-effective and adaptive parking system to address the problems faced by a driver when trying to find a free parking space in an urban environment. Unlike existing approaches, FPS assigns parking spaces to drivers in a way that optimizes the social welfare. To minimize the total travel time for all drivers, FPS uses a novel parking assignment algorithm, FPA, to assign each driver to an available parking space close to her destination in a way that reduces the total travel time (*i.e.,* the sum of the driving time to the parking space and the walking time from the parking space to the destination). FPA manages the effect of unsubscribed drivers that compete with FPS drivers for parking spaces. FPS was tested on a real road network and compared to Greedy and Naive parking assignment algorithms. The results show significant performance improvement over the other systems.

CHAPTER 4

DISTRIBUTED FREE PARKING ASSIGNMENT

The proposed parking assignment system in Chapter 3 is centralized, which makes it a bottleneck, as the server has to perform intensive computation and communication with the drivers, and a privacy risk, as the drivers have to disclose their destinations to the server. To address these limitations, this chapter presents DFPS, a distributed mobile system for free parking assignment. The chapter starts with system model of DFPS together with its scalability and privacy goals in Section 4.1. Section 4.2 describes the entropy-based cloaking technique to conceal drivers' destinations in the parking requests. Subsequently, the K-D tree structure of the cooperative drivers is explained in Section 4.3. Section 4.4 presents the parking assignment algorithm. The privacy analysis and performance evaluation are shown in Section 4.5 and Section 4.6, respectively. The chapter concludes in Section 4.7.

4.1 System Overview

This section presents an overview of DFPS, with emphasis on its design goals and system/threat models.

4.1.1 Design Goal

Our design goal is to propose a solution that solves two intrinsic problems in a centralized system for parking assignment: scalability and privacy. In a centralized system, the server responsible for communication with drivers and parking request processing could be a bottleneck. Also, processing parking requests requires drivers to disclose their desired destinations to the server. This could lead to major privacy concerns for the drivers and may prevent this solution from being adopted. Thus, the following system objectives should be considered to achieve the design goal:

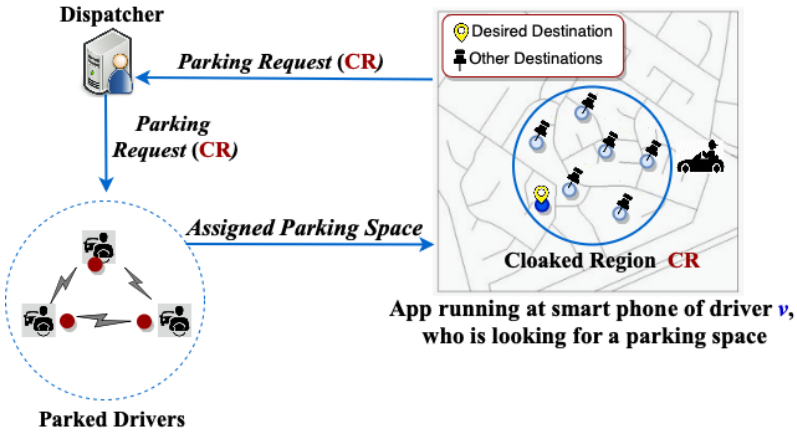

Figure 4.1 DFPS system architecture.

- Scalability: The parking assignment process should be distributed and performed efficiently on the drivers' smart phones rather than at the central server to reduce the computation and communication burden on the server and to achieve better scalability.

- Privacy: The drivers' destinations should be protected. When a driver submits her parking requests through the central server, the server cannot identify her desired destinations and cannot link different parking requests submitted by the same driver at different times.

4.1.2 System Model

Under the aforementioned system objectives, we propose the following system model to implement DFPS (Figure 4.1). The system consists of the following two entities: a parking app running on the drivers' smart phones and a parking assignment dispatcher running on a server.

- **Drivers** in the system are divided into three categories, based on their status: (1) drivers who are looking for parking spaces, (2) parked drivers, and (3)

departing drivers. The drivers in the first category determine their cloaked regions, submit their requests along with the regions, and then wait to receive parking spaces while heading to their destinations. Once parking assignments are made, they drive to the parking spaces. When a driver has parked (second category), she cooperates with other parked drivers to provide on-demand parking service for the drivers looking for parking spaces. Departing drivers report when they leave the system such that DFPS can update the status of the parking spaces. Drivers use their smart phones to communicate with the system and with other drivers through the Internet (*e.g.,* over 5G).

- **Dispatcher** is a cloud server that receives parking requests from the drivers looking for parking spaces, and forwards them to parked drivers who perform parking assignments. The dispatcher also serves as a bootstrapping unit to initialize the whole system. During the system's adoption period, when a few number of parked drivers provide the parking service, the dispatcher could share with the drivers parking availability information derived from historical parking statistics, real-time sources (*e.g.,* street images from video cameras), or real-time curbside parking occupancy data from sensors embedded in the smartphones of pedestrians, who walk on the sidewalks next to the parking spaces [60].

The high-level workflow of the DFPS architecture in Figure 4.1 illustrates the life-cycle of a parking request in the system, from generation to completion. When a request is generated, an entropy-based cloaking technique is used to protect privacy. Specifically, each driver has its own privacy profile that specifies the desired level of privacy. A privacy profile includes two parameters, a pseudonym and an integer k. The pseudonym ensures the pseudonymity of the parking request by concealing the driver's identity and k indicates that a driver wants her destination to be $k-$anonymous, *i.e.,* indistinguishable among $k-1$ neighbour destinations. In other

words, the driver wants to find a cloaked region that includes at least $k-1$ neighbour destinations to conceal her desired destination from the dispatcher. The larger the value of k, the more strict the privacy requirement is. The entropy-based cloaking technique generates cloaked regions (detailed in Section 4.2), which are sent along with parking requests.

After a parking request with a cloaked region is generated at the app on the smart phone, a question that arises is when the request should be sent and parking spaces are assigned. Assigning parking spaces when drivers are far away from their destinations increases the likelihood that the assigned spaces are taken by drivers who are not subscribed to DFPS (*i.e.,* unsubscribed drivers) and reduces the utilization of parking spaces. Assigning parking spaces too late may result in an increase in driving time and bad user experience, especially when the system may not be able to find a parking space close enough to the destination. Therefore, the cloaked regions serve a dual-purpose. As discussed, they provide privacy protection for drivers' destinations. In addition, DFPS uses them to determine when a driver's parking request has to be sent to the dispatcher: a parking request is sent to the dispatcher when the driver is about to reach the cloaked region.

In DFPS, the sizes of cloaked regions are determined by the following a few factors: 1) the need to preserve privacy: larger regions tend to contain more destinations and preserve privacy better; 2) the need for finding optimal parking spaces for drivers: with more available parking spaces, larger regions tend to provide more opportunities for DFPS algorithms to perform parking assignment optimizations; and 3) the need to avoid interference from unsubscribed drivers: with smaller regions, the contention of unsubscribed drivers for assigned parking spaces is less intense.

All the requests are sent to the dispatcher first. Then, the dispatcher forwards them to parked drivers. The work required to serve requests is divided among parked

drivers. This protects a driver's destination at the dispatcher side, minimizes the workload on the dispatcher, and maximizes the system scalability. We partition the whole area of a city into regions, and assign a parked driver to manage a region and allocate parking spaces in that region. We use a structured peer-to-peer network to organize parked drivers and the regions that they are in charge of. Once a driver parks in her assigned space, she is assigned to a region, joins the peer-to-peer network, and starts to serve parking requests from the drivers who are looking for parking spaces in her region.

4.1.3 Threat Model and Privacy Goals

Threat Model. We assume that the dispatcher is honest-but-curious, *i.e.,* it follows the protocol correctly, but may try to analyze the information available in the system in order to find private information about the drivers. For example, the dispatcher could be interested in learning personally identifiable information about the drivers such as their identity and their destinations. The dispatcher may also be interested in linking different parking requests made by the same driver, which could reveal over time private information about the drivers. We also assume that parked drivers do not collude with the dispatcher (*i.e.,* share information) or act maliciously in any other ways in order to break the privacy of other system users. Specifically, we do not handle the situation in which parked drivers may attempt to disclose the destination data of drivers for which they perform parking assignment.

A determined attacker could potentially be willing to expend resources and have a physical presence at a location in the real world in order to determine a driver's destination. We assume that the attacker's ability to execute such an attack is limited because it is expensive; whereas the attacker may be able to cover a small number of locations, it is not feasible nor cost effective to execute such an attack at scale.

The role of the dispatcher could be played by a telecommunication company which can also act as a service provider (*e.g.,* AT&T). This gives an unscrupulous dispatcher the chance to identify drivers' real identities as well as their destinations by tracking their location. In this work, we do not consider such a strong adversary.

Finally, we assume that the communication between entities in the system is protected using standard security mechanism against interference from external adversaries.

Privacy Goals. To mitigate the aforementioned privacy threats, DFPS needs to achieve the following privacy goals:

- *Driver identity privacy*: Drivers should not have to reveal personally identifying information (such as their real identity). This helps preserve the privacy of the drivers as related to their real-world persona, which may otherwise act as a deterrent against using the system.

- *Unlinkability of parking requests*: Given two parking requests, the system should not be able to tell if they are made by the same driver or by different drivers. This prevents building a profile of a driver over time, which may eventually lead to revealing a driver's real-world identity and their parking request history.

- *Parking destination privacy*: Given a parking request, the system should not learn the real destination of the parking request. This also prevents the system from learning information about a driver's real identity by correlation with the destination of parking requests.

There are well-known techniques to address the first two privacy goals, for example drivers can use a randomly chosen pseudonym for each parking request.

Our primary focus in this dissertation is to achieve the third goal, that of parking destination privacy.

4.2 Privacy-Aware Parking Request

Protecting driver's destination privacy against the attacks at the dispatcher side is one of the DFPS goals. Knowing the driver's destination could disclose sensitive information (*e.g.,* interests, the most visited places, etc.), which can cause privacy breaches, even if the driver uses a new pseudonym with every parking request [39, 47]. Privacy concerns in location-based services exist on two fronts: location privacy and query privacy [67, 68]. Spatial cloaking is a privacy protection mechanism used to protect both location and query privacy [50, 69]. The main idea is to blur a piece of location information by replacing the exact location with a cloaked region that contains the location and some other similar locations so as to satisfy the user's privacy requirement, *e.g.,* $k-$anonymity [39] (*i.e.,* the cloaked region contains at least k users). This mechanism is widely used because of its high efficiency. However, it must be improved to be usable in DFPS due to two drawbacks in our problem settings. First, it fails to consider the distribution of destinations, and a cloaked region with many destinations clustered together could be very small. Thus, the user privacy may be negatively affected. Second, it requires additional system resources from trusted third parties (*e.g.,* location anonymizer [50, 70], peer users [44]). This makes the system more complex and may bring additional vulnerabilities.

DFPS proposes an entropy-based cloaking technique that overcomes these drawbacks. DFPS generates cloaked regions that satisfy the privacy requirements of drivers by achieving $k-$anonymity. The construction of a cloaked region satisfies four requirements: (1) the cloaked region is not centered around the actual destination to avoid "center-of-cloaked-region" attack [43, 71]; (2) the destinations in the region are not very close to each other to avoid the clustering problem; (3) the region has enough

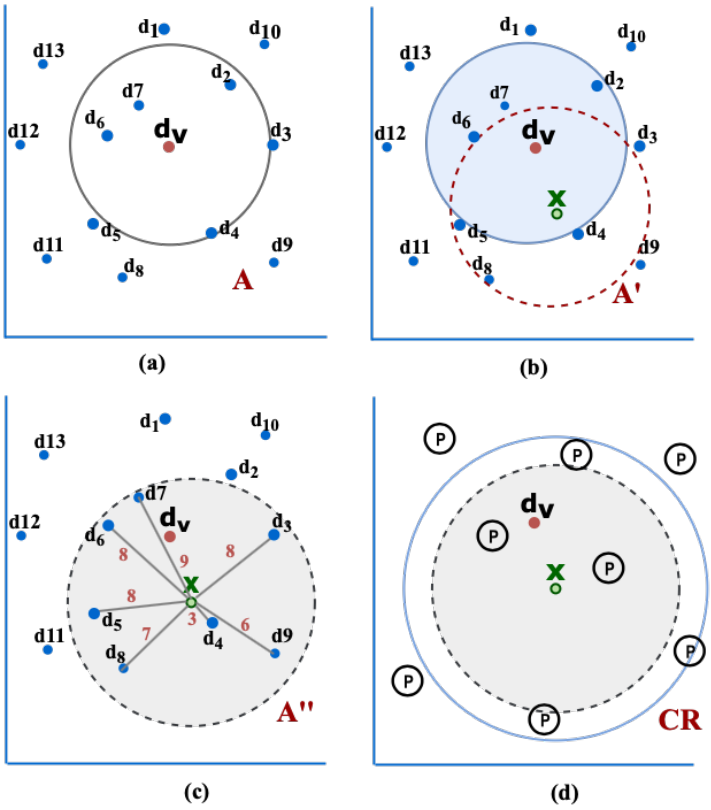

Figure 4.2 Entropy-based cloaking technique. (a) Basic k-anonymity region creation; (b) Center adjustment; (c) Entropy-distance adjustment; (d) Parking-availability adjustment.

open parking spaces to ensure the effectiveness of the parking assignment algorithm; and (4) the process of generating the region does not rely on trusted third-party servers.

These requirements are considered in the four phases of the cloaked region construction: (a) basic k-anonymity region creation; (b) center adjustment; (c) entropy-distance adjustment; and (d) parking-availability adjustment.

Figure 4.2 illustrates these phases, with a running example. In the figure, 13 destinations are represented with solid circles, and d_v is the real destination of the driver. We assume that the required $k-$anonymity level is four, *i.e.*, $k = 4$. The operations in these phases are as follows:

Phase (a): Basic k-anonymity region creation: This phase starts by applying the $k-$nearest neighbours algorithm (KNN) to determine the $k-1$ nearest destinations to d_v. Then, as shown in Figure 4.2(a), it defines a circular region A centered at d_v that encompasses the $k-1$ nearest destinations (*i.e.,* $NumDest(A) \geq k$).

Phase (b): Center adjustment: For the different requests with the same destination, phase (a) always generates the same cloaked region. This makes the solution vulnerable to "center-of-cloaked-region" privacy attack — an attacker could guess that the destination closest to the center of the cloaked region is the real destination of the driver. Thus, we propose an adjustment scheme for the region's center. As shown in Figure 4.2(b), DFPS randomly selects a point x in A and finds a set of $k-1$ nearest destinations to x. Phase (a) guarantees that x is not too far from d_v. DFPS then constructs a new region A', which is centered at x and contains the k-nearest destinations (including the real destination d_v).

Phase (c): Entropy-distance adjustment: Region A' satisfies requirement (1), but not requirements (2) and (3). It is possible that the distance between the $k-1$ neighboring destinations in A' is small, making it easier for an attacker to infer the driver's destination. Knowing the driver's identity helps an attacker to physically inspect which destination the driver is located at. To prevent this type of attack, the cloaked region may be further expanded while keeping x as its center. This adjustment is conducted using the entropy of distance method [18]. If the destinations of A' are located on fewer than k/2 segments (*i.e.,* they are too close to each other), DFPS selects a different set of $k-1$ neighboring destinations and ensures that: (i) the selected $k-1$ destinations are evenly distributed in the new, expanded region; and (ii) the new region is not expanded too much.

We first use the KNN algorithm to find $2k$ nearest neighbour destinations around x. The area containing all these destinations is inevitably large. Thus, we examine

the distribution of these $2k$ destinations to determine a smaller region A'', which contains $k-1$ destinations that can hide well the location of the real destination, based on the entropy values defined below. The number $2k$ is experimentally determined and provides a good trade-off between performance and overhead. In a practical deployment, DFPS can allow users to adjust a threshold for this method (in addition to the anonymity parameter k).

DFPS selects the $k-1$ destinations in this phase such that these destinations have a large entropy value, indicating that they are evenly distributed in the region. For this, DFPS forms m destination groups, each of which has $k-1$ destinations randomly selected from the $2k$ destinations. For each group, DFPS calculates two values: 1) an aggregated distance D, which is the sum of the distances between each of the $k-1$ destinations and the center x, and 2) an entropy value H, which measures the uncertainty of the destinations (more uncertainty indicates more even distribution). The entropy of a group is calculated using the following equation.

$$H(n) = -\sum_{i=1}^{k-1} \propto_{n_i} \log \propto_{n_i} \tag{4.1}$$

In the equation, $\propto_{n_i}$ is the the weight of the destination i, which is defined as follows:

$$\propto_{n_i} = \frac{dist(x, d_i)}{\sum_{j=1}^{2k} dist(x, d_j)} \quad (i = 1, .., k-1) \tag{4.2}$$

Among m groups, we select the group to determine the cloaked region A'' as follows. If the D values of the groups are equal, we select the group whose destinations are more evenly distributed (larger entropy value). Otherwise, we select the group with the largest D value.

Figure 4.2(c) depicts how a cloaked region is expanded using the entropy of distance method. The area of A' is first expanded to cover $2k$=8 nearest destinations including d_v. Lines between the destinations indicate the spatial neighbor relation between each destination and x, and the values marked on the lines indicate the distances. Assume that three (m=3) groups of destinations are randomly selected, G1=$\{x, d_5, d_8, d_9\}$, G2=$\{x, d_3, d_5, d_6\}$, and G3=$\{x, d_5, d_7, d_8\}$. According to formula 4.1, the entropy on distances is obtained. The total distance of destinations is also calculated for each group. The total distance of destinations in $G1$ is less than those of $G2$ and $G3$. The total distances of $G2$ and $G3$ are almost equal; however, the destinations in $G2$ are more evenly distributed than the ones in $G3$. According to the entropy of distance method, $G2$ is chosen in this example. Then, the cloaked area A'' is computed based on group $G2$. The cloaked area shown in the figure is before the shrinking done according to $G2$.

Phase (d): Parking-availability adjustment: this phase happens when the DFPS app at the driver sends a parking request for region A'' to the dispatcher. As explained, A'' guarantees k-anonymity destination protection (*i.e.,* the dispatcher cannot tell the real destination from $k-1$ other destinations, which are not clustered together). Upon receiving the request, the dispatcher has to decide whether to forward it to the peer-to-peer network of parked drivers as is or to expand the region further. The decision is based on the parking availability in A''. The dispatcher dynamically maintains the spatial distribution of parking availability in the whole city as we will describe in Section 4.3.5. If the number of available spaces in A'' is less than a threshold P_{min}, the dispatcher expands the region to encompass the closest available parking spaces to center x that are not within A'' until the number of parking spaces in the region reaches P_{min}. This is done to avoid the parking space scarcity problem. In DFPS, there is always a chance that an unsubscribed driver will take a parking space before the driver assigned to that space arrives there. The probability of this

situation to happen is much higher when there is parking space scarcity in a region. Expanding the region up to P_{min} parking spaces reduces the likelihood of such a situation.

Figure 4.2(d) depicts how the cloaked region A'' is expanded to satisfy the parking availability condition. We assume that $P_{min} = 3$, and the number of available parking spaces within A'' is two. A'' will be expanded to contain at least three available parking spaces.

The Dispatcher over-estimates the number of available parking spaces as it uses only information from the DFPS parked drivers. This is because unsubscribed drivers may take parking spaces presumed to be available by our system (this problem is addressed in Section 3.4 based on keeping track of spaces occupied by unsubscribed drivers and on avoiding assigning these spaces for a period of time). Thus, the value of P_{min} should be reasonably large to tolerate this over-estimation. This value is determined experimentally as a function of the number of destinations and the total number of spaces in the region in order to provide a good trade-off between destination privacy and system optimization.

An alternative solution that avoids phase (d) is to make the cloaked region significantly larger than A'' before submitting the request. In this way, there will be a high chance to find parking spaces available in the region. However, this alternative solution may assign the parking spaces too early (when the drivers are far away from their parking spaces), and the parking spaces may be taken by unsubscribed drivers before the subscribed drivers arrive there. Thus, we choose to apply phase (d) instead of this alternative solution.

4.3 Overlay Network Structure and Operation

DFPS uses a peer-to-peer overlay network to organize parked drivers and the regions managed by the parked drivers. The organization of parked drivers needs to satisfy the following requirements:

- *Scalability*: The drivers must be organized in a scalable way to share the workload effectively.

- *Fast Routing*: Given a request, DFPS must quickly identify the driver managing the region where the parking space is requested.

- *Adaptability*: The overlay network must adapt quickly and with low overhead to high churn (*i.e.,* parked drivers coming and going frequently).

4.3.1 K-D Tree Network Structure

To meet the above requirements, DFPS organizes the overlay network of the parked drivers as a K-D tree. A K-D tree is a k-dimensional binary search tree for partitioning and spatially indexing data distribution in a k-dimensional space [15, 72]. A node in the K-D tree is associated with three types of information: a value, a rectangular representation (*i.e.,* a region) containing a set of data points, and the coordinates of these data points.

Each node in the K-D tree represents a region. The region corresponding to a parent node is divided into sub-regions corresponding to the children of that node. The locations of the parking spaces in a region are represented as data points associated with the node for that region. The node's value is the location of the first driver parked in the corresponding region when the region and the node are created. For brevity, we call this value the location of the node.

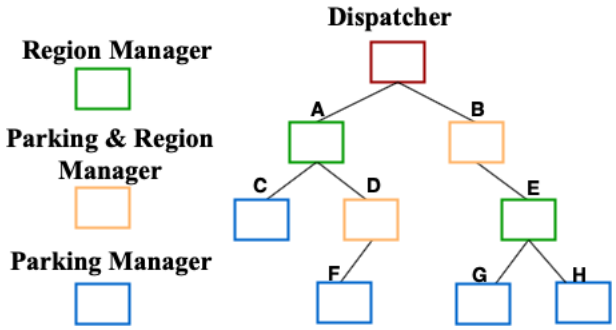

Figure 4.3 The roles associated with nodes in the K-D tree.

DFPS associates a parked driver with each tree node. The tasks of forwarding parking requests and allocating parking spaces, as well as the data structures required to manage these tasks, are associated with nodes. Since the nodes follow a tree structure, DFPS can manage the tasks and data structures in a hierarchical way, which leads to good scalability.

There are two roles that may be associated with a node in the K-D tree as shown in Figure 4.3: 1) *region manager* which forwards parking requests, divides a region into two sub-regions when necessary, and assigns sub-regions to drivers that park in these sub-regions; 2) *parking manager* which assigns parking spaces within the region associated with the node. Depending on its position in the tree, a node may have one or both of these roles. A leaf node acts only as parking manager for its region (*i.e.,* nodes C, F, G, and H). An internal node that has two children acts only as region manager (*i.e.,* nodes A and E). An internal node that has only one child acts as parking manager for the sub-region that is not covered by the child node, and it acts as a region manager by forwarding requests to its child or by assigning the sub-region not covered by the child to a driver that parks in that sub-region (*i.e.,* nodes B and D).

Since parking space allocation is handled by the phones of parked drivers, we also refer to the parked drivers as the region manager or the parking manager of the regions corresponding to the node (depending on the node's role).

4.3.2 Joining and Departing K-D Tree

The K-D tree grows dynamically when more drivers park. When a driver informs its parking manager that it had parked, the parking manager creates a sub-region and a new node for the sub-region. Then, it attaches the new node as the child of the node it manages and assigns the newly parked driver to manage the new node and the parking spaces in the corresponding sub-region. Thus, the newly parked driver becomes the parking manager for these parking spaces. Over time, it also becomes a region manager when it has to divide this sub-region.

When a parking manager creates sub-regions, it divides its region based on the location of its parking space. This design has two advantages over evenly splitting the entire region among all the parked drivers. First, it helps to evenly distribute parking requests to parking managers. Due to hot spots, the destinations of drivers are not distributed evenly in the region. If the entire region is evenly partitioned among parking managers, the drivers managing hot areas might be overloaded. In contrast, the method employed by DFPS guarantees that more parking managers are assigned to hot areas than cold areas. Second, sub-regions are created dynamically within a small region where the driver parks. Other regions are not affected. This minimizes the changes to regions and the associated overhead, such as exchanged messages.

Figure 4.4 shows an example illustrating how a K-D tree grows, in which four drivers (A, C, B, D) park sequentially in a region with a size of 8x8. Figure 4.4(a) shows how the sub-regions are created when these drivers park, and Figure 4.4(b) shows how each new node is created and added to the K-D tree. Initially, before any driver parks, there is only one node (*i.e.,* the dispatcher) in the tree, managing the

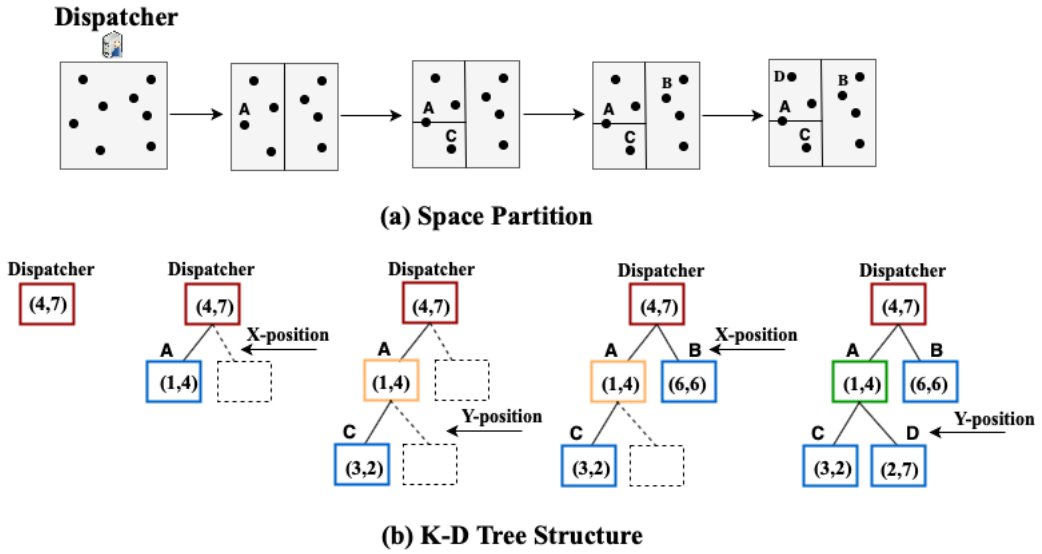

Figure 4.4 Example showing how a K-D tree grows when drivers A, C, B, D park in a 8x8 area. Dots represent parking spaces and letters represent parked drivers. The numbers in each box of sub-figure (b) are the 2-D coordinates of the parking space of the corresponding driver.

entire region. The location of the dispatcher ((4,7) in the figure) is chosen such that its value on the x-coordinate is in the middle of the region, whereas its value on the y-coordinate is chosen at random. Choosing the x value in the middle helps with system load balancing, as the first split of the space in the K-D tree is done on the x-coordinate.

When driver A parks at coordinate (1, 4), the whole region is split into two sub-regions. Sub-regions are created by splitting the parent region along alternating dimensions depending on the K-D tree level of the node managing the parent region. At the root level, the x-dimension is used. In our example, the first splitting is along the x-coordinate of the dispatcher. The sub-region where driver A parks is assigned to A. Although the dispatcher is still the region manager of the entire region, node A is the parking manager of the sub-region assigned to it. The dispatcher remains the parking manager for the other sub-region.

Driver C requests a parking space in the region whose parking manager is A. Assume that A assigns parking space (3,2) for driver C. When C parks, the new node created for C is added as a child node of A. Thus, A splits its region into two sub-regions based on the y-coordinate of A's parking space location. Since the parking space of C is in the bottom sub-region, C becomes the parking manager of this sub-region. Although A is the region manager of the region consisting of the two (top and bottom) sub-regions, it acts as the parking manager only for the top sub-region.

Driver B requests a parking space in the region managed by the dispatcher, and the dispatcher allocates a parking space at (6,6) to B. When B parks, the node created for B is added as a child of the root. It then becomes the parking manager of the dispatcher's second sub-region (the right half of the whole region). Since A and B handle the parking space allocation of both the dispatcher's sub-regions, the dispatcher no longer handles any parking space allocation.

When parked drivers leave their parking spaces, the tree nodes associated to those drivers must be updated. For each node managed by a leaving driver, if the node is a leaf, the node is deleted from the K-D tree and its parent node (*i.e.,* the corresponding driver) takes over the parking space allocation task associated with the node. If the node is an internal node, one option is to apply existing solutions for deleting K-D tree internal nodes [72]. However, because the sub-trees rooted at the internal node's children must be re-organized to form another sub-tree, these solutions can be very expensive and may cause considerable overhead, especially when the sub-trees are large.

Instead of deleting an internal node, DFPS assigns the node to another parked driver. DFPS considers the following two situations:

- The node is the parking manager of the region containing its location (the node value). In this case, the node has only one child node. DFPS will find the driver who is managing the child node, and assign the node to this driver.

- The location of the node is managed by another node, *i.e.,* another node is the parking manager of the region containing its location. DFPS will first locate the parking manager, then assign the node to the driver of the parking manager. In the example shown in Figure 4.4, when A leaves, driver D will be asked to take care of the node of A, since D is in charge of the parking space allocation in the region where A parked. Thus, later on, when the parking space is re-allocated by D to another driver E, E can be assigned to manage the node.

4.3.3 Request Forwarding

The same overlay network can be used to forward the parking requests for two types of drivers: (a) drivers who care to set their privacy preferences, and (b) drivers who do not require privacy. The second type of drivers may end up with parking spaces closer to their destinations. In the following, the parking request forwarding procedures explain how a request is forwarded to the corresponding parking manager with and without privacy.

Forwarding with Privacy. Each parking request, which includes the pseudonym of the driver and the cloaked region computed at the driver's phone, is forwarded down the tree from the root (*i.e.,* dispatcher) until it reaches the corresponding parking manager, which will assign a parking space in its region. This process is described in Algorithm 2.

Upon node n receiving a parking request along with the cloaked region CK, n's state is examined to determine if it can answer this request or forward it down the

Algorithm 2 Parking request forwarding procedure

1: **Upon node** n **receiving a parking request (**$v_{pseudonym}$**,** CK**)**
2: **if** (n is not a *parking manager*) **then**
3: Forward the request to the children whose regions intersect with CK
4: **else if** (n has no children) **then**
5: //Region(n) is the region managed by n when acting as parking manager
6: Send the coordinates of Region(n) to the driver
7: **else**
8: //n is a parking manager and a region manager
9: **if** $(CK \cap Region(n))$ and $(CK \cap Region(child(n)))$ **then**
10: Send the coordinates of Region(n) to the driver and forward the request to the child
11: **else if** $(CK \cap Region(n))$ and $(\neg(CK \cap Region(child(n))))$ **then**
12: Send the coordinates of Region(n) to the driver
13: **else**
14: Forward the request to the child node
15: **end if**
16: **end if**

tree. If n is a region manager, then the request is forwarded to the children whose regions overlap with CK (lines 2-3). CK could overlap the region of one child or both. If n is a parking manager and its region overlaps with the CK, then the region coordinates are sent to the driver (lines 4-6). If n is both a parking manager and a region manager, then both its region and its child's region have to be examined (lines 7-8). If CK overlaps with both regions, n sends the coordinates of its region to the driver and forwards the request to the child (lines 9-10). If only n's region overlaps with CK, n sends the coordinates of its region to the driver (lines 11-12). Otherwise, it forwards the request to the child (lines 13-14).

This process works recursively until the coordinates of all regions that intersect with CK are sent to the driver, along with the identities of their parking managers. The app at the driver determines in which region her destination is located and then communicates directly with the parking manager of that region. In this communication, the app at the driver requests a parking space from the parking manager using its exact destination, not the cloaked region. The parking manager then performs parking assignment, as described in Section 4.4. In this way, only one

parking manager learns the driver's destination. All the other nodes in the tree that processed the original request know only the cloaked region CK. This brings some level of privacy for the drivers since non of the parked drivers have access to the their exact destinations except the one who is in charge of providing the parking service for them.

Forwarding without Privacy. When a driver does not require privacy, she submits her parking request with her driver's id and the desired destination only. The request is sent to the dispatcher in order to forwarded it to the corresponding parking manager as described in Algorithm 3.

Algorithm 3 Parking request forwarding procedure

```
1: Upon node n receiving a parking request (v, d_v)
   // v refers to a driver's identity and d_v represents the driver's destination
   // Region(n) is the region managed by n when acting as parking manager
2: if (n is not a parking manager) then
3:    Forward the request to the child whose region contains d_v.
4: else if (n is a parking manager that has no children) and (d_v ∈ Region(n)) then
5:    Accept the request.
6: else if (n is a parking manager that has one child) then
7:    if (d_v ∈ Region(n)) then
8:       Accept the request.
9:    else
10:      Forward the request to the child node.
11:   end if
12: end if
```

Upon node n receiving a parking request, the state of node n is examined. If n is a region manager, the request is forwarded to its child whose region contain d_v (line 2-3). The request is accepted if n is a parking manager and d_v is located in its region (lines 4-5). If n is a region manager and parking manager, then both its region and its child region have to be examined (lines 6-11). The request is accepted if n's region contains d_v; otherwise, the request is forwarded to its child.

4.3.4 Load Balancing

Each parking manager receives requests for its region. However, the distribution of destinations and requests coupled with the tree-structure of the network can cause a heavy load on certain managers. Heavy load leads to slow downs and significant battery consumption on the impacted phones. To avoid this problem, DFPS applies a simple load balancing mechanism. An overload threshold is determined by each parking manager as the difference between the local load δ (*i.e.,* the number of requests that have been processed by the phone) and a load threshold α. If $\delta > \alpha$, an imbalance is detected and the phone removes itself from the system. The threshold α can be determined experimentally on each phone such that the phone does not consume more than a small fraction of its battery power serving DFPS requests. As described in Section 4.3.2, a phone of another parked driver will replace this phone in the overlay network and will handle any pending requests inherited from the removed phone.

4.3.5 Failure Recovery

DFPS employs the proposed mechanism in Midas [72] to ensure that the system continues to function in the presence of failures or disconnections of the phones of parked drivers. For example, the phones may fail without warning if the drivers decide to turn them off. Failure or disconnection of the phones is detected by periodic gossip messages from their neighbors. Each neighbor knows the region boundary of the failing node w and maintains a replica of the data it stores (*e.g.,* the number of available spaces and total number of spaces) in order to restore the data and improve availability. In addition, a parent maintains a list of requests forwarded to w and requests assigned by w in case of failure. To avoid consistency problems, a disconnected parking manager will not attempt to reconnect to the system when the wireless connection becomes available again. Finally, let us note that the overload

threshold at parking managers (used for load balancing as described in Section 4.3.4) also reduces the effect of node failures or disconnections.

4.4 Parking Space Assignment

In DFPS, each parking manager periodically runs the same parking space assignment algorithm to satisfy the outstanding requests that have been forwarded to it. The algorithm computes an assignment for these requests, aiming to optimize the total travel time of the drivers.

4.4.1 Parking Space Assignment Algorithm

Each parking manager in DFPS assigns parking spaces in its region in the same way as the dispatcher assigns parking spaces in our centralized solution. The detailed algorithm can be found in Section 3.4. A brief description is included below for convenience.

A parking manager assigns parking spaces periodically to outstanding requests. Each manager can adjust the period as a function of its outstanding request queue size to achieve a good trade-off between assignment performance and overhead. In each period, the manager first pre-allocates to the driver of each outstanding request the closest available parking space to her destination. The pre-allocation adapts the solution to the flow problem described in the Parking Slot Assignment Games (Psag) [4] to minimize the total walking time of these drivers.

The actual assignment of parking spaces takes place based on the urgency of the demands for parking spaces (*i.e.,* how close the drivers are to their destinations). We apply a modified version of the compound laxity algorithm to determine how long a request can be delayed before it must be assigned a space. Specifically, in each period, the drivers with the most urgent demand are selected. Their pre-allocated parking spaces are officially allocated to them.

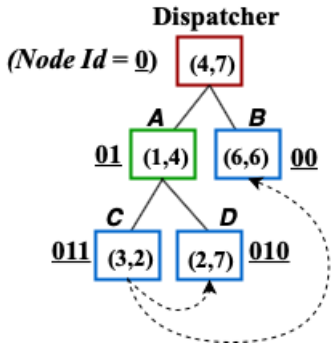

Figure 4.5 Nodes in the maximal sibling sub-trees of node C.

Table 4.1 Neighbor Relation Table

Node	Neighbors		
$C(011)$	$D(010)$	$B(00)$	
$D(010)$	$C(011)$	$B(00)$	
$B(00)$	A (01)	$C(011)$	$D(010)$

The algorithm described above is named FPA. It assumes that subscribed drivers are generally representative of the entire driving population and all spaces in the region are available to them. To consider the cases in which spaces may be taken silently by unsubscribed drivers, the algorithm is enhanced to track the spaces taken by unsubscribed drivers and avoid assigning these spaces. This algorithm, described in Section 3.4, is named FPA-1.

FPA and FPA-1 are used under the assumption that there are still available parking spaces in the region. However, in DFPS, the assignment of parking spaces is done by multiple parking managers. It is possible that a parking manager runs out of parking spaces, but still has outstanding requests. In such a case, DFPS allows a parking manager to acquire parking spaces from nearby parking managers temporarily to satisfy her outstanding requests, as explained next.

4.4.2 Finding Best Available Spaces from Neighboring Regions

DFPS uses an indexing scheme based on the multi-indexing technique in [72] to locate nearby regions close to the destinations of the parking requests. The scheme assigns a binary identifier to each node as its index. The binary format of the index reflects the path from the root to the node, and thus reflects roughly the location of the corresponding region. For example, as shown in Figure 4.5, the index of the root node is 0 and the indexes of its children nodes (A and B) are 01 and 00. The first bit of the indexes (*i.e.,* 0) reflects that they are in the region of the root node (*i.e.,* index 0). The second bit (*i.e.,* 1 or 0) reflects the corresponding sub-region created by the root node. Nodes C and D are the children of node A, and the first two bits (*i.e.,* 01) reflect that they are in the region of node A (*i.e.,* index 01), and the last bit reflects the sub-region.

In DFPS, each parking manager maintains a neighbor table, as shown in Table 4.1, which includes the nodes managing the neighboring regions, named neighboring nodes. For any two nodes (N_1 and N_2) with indexes of lengths L_1 and L_2 respectively, the two nodes are neighboring nodes if one of the following two conditions are met: 1) If $L_1 \leq L_2$, the first $L_1 - 1$ bits of the two indexes are the same, and the ${L_1}^{th}$ bits are different. 2) If $L_1 > L_2$, the first $L_2 - 1$ bits of the two indexes are the same, and the ${L_2}^{th}$ bits are different. The neighbor table is built by broadcasting the index of each newly-created node.

The best parking spaces are those close to the destinations of the requests. To find such spaces, a parking manager that runs out of spaces first needs to contact her neighboring nodes to get their lists of available parking spaces. Note that a neighboring node may not be a parking manager, which has first-hand information on available parking spaces. If a neighboring node is not a parking manager, to obtain a list of available parking spaces in its region, the node needs space availability reports from all its offsprings. Then, the parking manager short of spaces examines

the parking spaces in the lists received from its neighbors, calculates the distances between the spaces and the destinations of the pending requests, and selects the parking space with the shortest distance for each of these requests.

The multi-indexing technique is also leveraged to pass the parking availability information up the tree to the dispatcher, which needs it for phase (d) of the cloaked region construction.

4.5 Privacy Analysis

This section analyzes how the DFPS protocol satisfies the desired privacy goals.

Driver Identity privacy: A driver uses a randomly-generated pseudonym identity with every parking request, which is completely unrelated to her true identity. This pseudonym is not identifiable by attackers at the dispatcher side because (1) the parking request is a snapshot query (*i.e,* a request submitted just once by the driver); (2) the pseudonymity mechanism is effective due to the discrete characteristic of the parking behavior (*i.e.,* the average time interval between two parking demands is long enough). Thus, the attacker cannot infer the driver identity from a parking request.

Unlinkability of parking requests: Given two parking requests, no one should be able to tell if they are made by the same driver or by different drivers. This is achieved by the use of pseudonyms and cloaked regions. In other words, with each parking request, a driver's privacy is protected by (1) replacing her true identity with a randomly-generated pseudonym; (2) constructing the cloaked regions such that two requests from the same driver to the same destination will result in different cloaked regions.

Parking destination privacy: By design, the cloaked area contains k destinations, which ensures the driver's true destination is hidden among these k destinations. However, to infer a driver's destination, the attacker may deploy a "center-of-cloaked-region" privacy attack [43, 71], *i.e.,* the destination is inferred to

be at the center of the cloaked region. The attacker may also be physically present at a location to determine a driver's destination.

The parking request in DFPS includes provisions to mitigate these attacks. To alleviate the "center-of-cloaked region" attack, phase (b) of the parking request protocol randomly shifts the center of the cloaked region. Thus, even if the same driver chooses to travel to the same destination on different occasions, the cloaked area (which contains the true destination) will appear differently to the dispatcher.

Although the cloaked area contains k destinations after phase (b) of the parking request protocol, it may still be possible that these k destinations be clustered in a small region. If the attacker decides to be physically present in this small region, the driver's true destination may be determined based on direct observation. To mitigate this issue, phase (c) of the parking request protocol uses the entropy of distance method to select a cloaked region which contains k destinations that are located on more than k/2 segments and are evenly distributed in the cloaked region. As a result, the probability of inferring the true destination within the cloaked region remains $1/k$.

Due to the fact that that dispatcher manages the entire space before any driver parks, the real destinations of the first right and left managers will be known, which can lead to privacy leakage. However, this has a very minor effect on the privacy of the whole system.

4.6 Experimental Evaluation

In this section, we experimentally evaluate the performance of DFPS, in terms of (1) scalability and load balancing, (2) assessing the benefits of DFPS with and without privacy compared to the centralized system; (3) measuring the benefits of DFPS in terms of travel time when compared to a Naive parking assignment solution, which resembles what drivers normally do; (4) investigating the DFPS performance under

different privacy protection mechanisms; (5) understanding the relationship between the average travel time and the size of the cloaked regions; (6) analysing the impact of the privacy level on the average travel time.

The evaluation is done via simulations over a real road network. The experiments simulate two different scenarios: subscribed-drivers-only scenario, which assumes that all drivers in the system use DFPS; unsubscribed-drivers-interference scenario, which assumes there are a number of drivers who have not subscribed to DFPS and who may occupy, without notification, parking spaces known to the system as available.

We use SUMO/TraaS [64], an open source framework for running vehicular network simulations, and PeerSim [73], a simulation environment for P2P protocols. SUMO/TraaS simulates vehicles going to their destinations in a business district in Manhattan, New York City (see Figure 3.5). PeerSim simulates the overlay peer-to-peer network of parked drivers.

We generate a set of trips for the drivers. While the starting locations of the vehicles are randomly chosen over the entire road network, the target destinations are chosen randomly from a small region in the center of the map to ensure enough contention for parking spaces. The travel time of a driver is the sum of the driving time and the walking time. In these experiments, the driving time is the sum of the time the driver takes to reach the cloaked region and the time from the edge of the cloaked region to the assigned parking space. The walking time is calculated from the assigned parking space to the destination. We consider a walking speed of 1.4 m/s [65]. DFPS starts each test with 1024 vacant parking spaces. The arrival rate of the requests falls within the range of 1 to 5 requests per second. For each experiment, we collect results from 5 runs and average them.

We compare the performance of DFPS with (1) our centralized system FPS in Chapter 3 and (2) a version of DFPS without privacy protection (DFPS-wop) in terms

of the average travel time. Unlike DFPS, which uses the edge of the cloaked region to trigger a parking assignment request, FPS and DFPS-wop trigger the request when the driver reaches a circle centered at the destination and with a radius (request distance) set based on the parking availability in that region. The radius is initially set to the average length of the road segments within the whole region, and it is adjusted periodically based on the parking occupancy rate in the region: the radius is increased when the occupancy becomes higher. Each driver in DFPS selects the value of her $k-$anonymity randomly from the set {3,5,7,9}.

4.6.1 Results and Analysis

Scalability and Load Balancing. Compared to our centralized system FPS, the computation bottleneck at the central server (*i.e.,* dispatcher) in DFPS is removed, as the parking assignment is computed in a distributed fashion by the phones of the drivers. Therefore, DFPS scales better from a computation point of view.

The total computation time consumed in each period for the parking assignment algorithm is the sum of (1) finding a valid allocation that minimizes the total walking time to destinations and (2) determining the urgency of pending requests in order to assign spaces to the most urgent requests and minimizing the total driving time.

Minimum-cost network flow in a directed bipartite graph is used to generate a valid allocation. Its cost is $O(ve)$, where v is a number of nodes (*i.e.,* m spaces + n drivers in the region) and e is the number of edges (*i.e.,* equal to n, the number of drivers in the region). The cost of computation to determine request urgency and select urgent requests is $O((n+\varphi)^2)$, where n is the number of drivers to be assigned parking spaces and φ is the number of drivers with high urgency. Thus, the total time for each parking manager is $O(n^2+nm)+O((n+\varphi)^2)$. Given that each parking manager handles only a limited number of parking spaces and drivers, as described in Section 4.3.4, this computation can easily be done on today's smart phones.

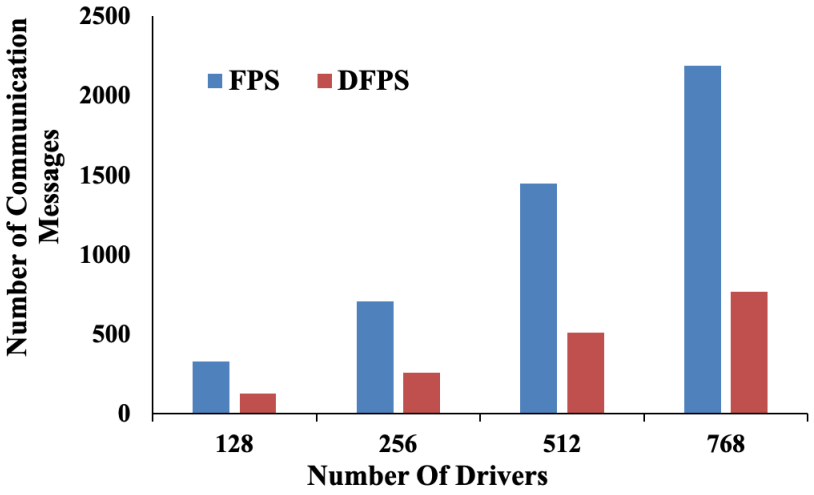

Figure 4.6 Number of messages handled by the dispatcher/server in DFPS and FPS for a different number of drivers and a fixed number of destinations (8).

Figure 4.6 compares the number of messages handled by the dispatcher in DFPS and the server in FPS by varying the number of drivers from 128 to 768. The results show that DFPS reduces the number of messages by a factor of 2 or better when compared to FPS.

Next, we investigate the effect of load balancing on parking managers. We compare the number of requests assigned by the phone of a parked driver when DFPS employs its load balancing mechanism (DFPS) and when it does not (DFPS*). The value of the load threshold α in the load balancing mechanism is set to 10. Table 4.2 compares the maximum number of assigned requests in DFPS and DFPS*. As expected, DFPS scales better due to its load balancing mechanism. The maximum number of requests in DFPS* is about 20 times higher than the maximum number in DFPS. We also observe that the maximum in DFPS is 13, not 10 as expected (the load threshold). This is because of two reasons. First, a parking manager receives requests until it has served α of them (while the others are pending). Second, a parking manager can not depart the network until it completes its set of requests with high urgency.

Table 4.2 Maximum Number of Requests Assigned by a Parked Driver for Different Numbers of Destinations and 768 Drivers

	Destinations		
	2	**4**	**8**
DFPS	13	12	12
DFPS*	268	166	100

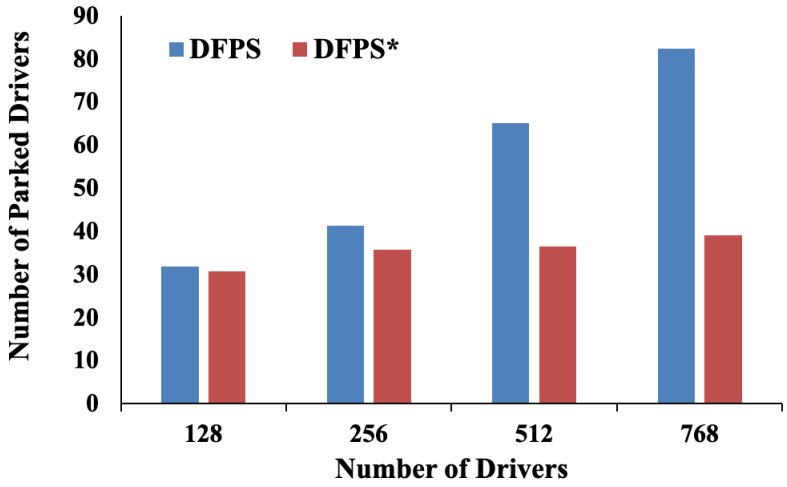

Figure 4.7 Number of parked drivers involved in the assignment process for different total numbers of drivers and eight destinations.

Figure 4.7 presents another measure of scalability: the average number of parking managers cooperating to serve the incoming parking requests in DFPS and DFPS*. The results show that the number of managers in DFPS increases by as much as 185% compared to DFPS*. Higher numbers are better because the load is distributed more evenly across the participants, and the system scales better.

Average travel time. The average travel time measures the performance of the system from a global point of view. Figure 4.8 shows the average travel time for DFPS compared to FPS and DFPS-wop in the *subscribed-drivers-only* scenario. In FPS and DFPS-wop, destination privacy is not considered. Their regions are constructed based only on parking availability. The results show that DFPS reduces the travel time by

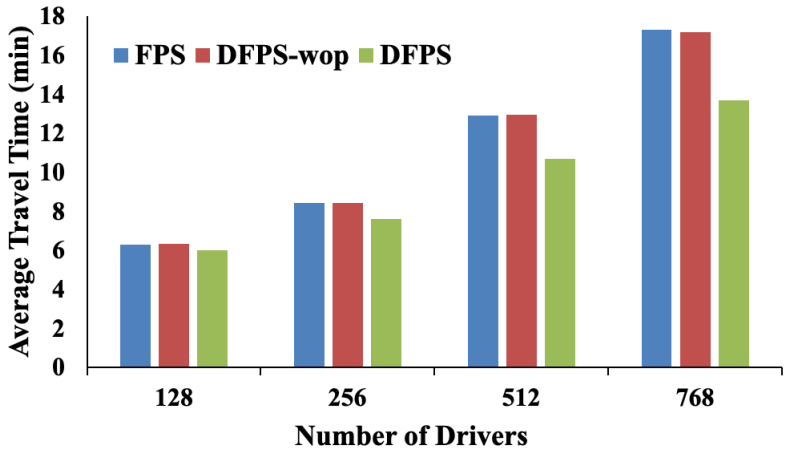

Figure 4.8 Average travel time for a different number of drivers and eight destinations.

as much as 26% compared to DFPS-wop and FPS. We observe that a DFPS-wop and FPS have similar results.

The improvement in performance observed when comparing DFPS with DFPS-wop and FPS is due to combining the privacy and parking availability requirements when generating the cloaked region. Let us recall that DFPS-wop and FPS use just parking availability to generate their regions. The privacy requirement allows DFPS to optimize the size of the region better than DFPS-wop and FPS. This is due to the even distribution of the destinations, which also distributes better the available parking spaces. An optimized region allows for more effective parking assignment optimizations.

For the same scenario, Figure 4.9 shows the average travel time when the number of destinations is varied from 2 to 8. The figure breaks down the travel time into two parts: driving time and walking time and shows that drivers spend most time on driving, and DFPS reduces the average travel times by mostly reducing the driving time. With eight destinations, DFPS can reduce driving time by 67% and 64% compared to FPS and DFPS-wop, respectively. Reducing the driving time is very

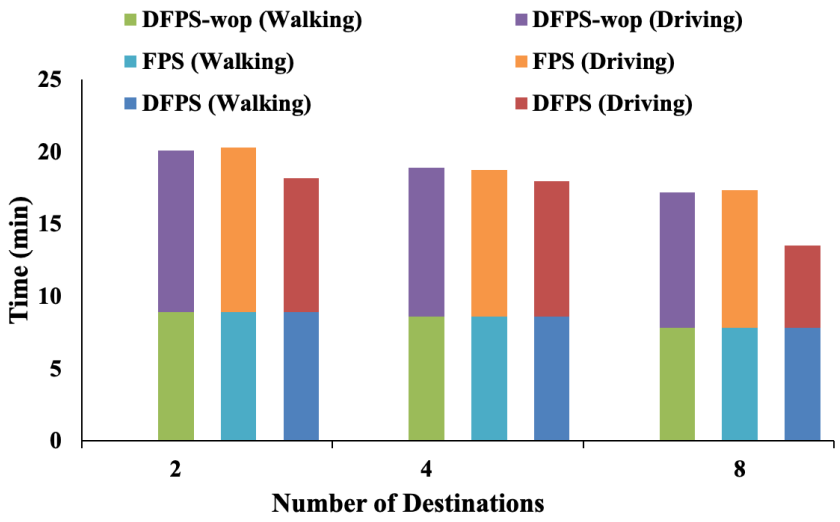

Figure 4.9 Walking time and driving time for different number of destinations and 768 drivers.

important, as this reduces traffic congestion and implicitly the gas cost and pollution. Since the number of parking spaces in the centroid area is limited, all systems can hardly reduce the walking time.

The next set of experiments evaluate the performance of DFPS and DFPS-wop in the *unsubscribed-drivers-interference* scenario. Each system, has two parking assignment algorithms. One (DFPS/FPA and DFPS-wop/FPA) just keeps trying to find another space if the assigned parking space is found to be taken by an unsubscribed driver. The other (DFPS/FPA-1 and DFPS-wop/FPA-1) keeps track of the spaces found to be taken by unsubscribed drivers, avoids them for a while, and tries them later. We call the spaces taken by unsubscribed drivers "hidden" spaces. These spaces are taken at the beginning of the simulation to help tracking them. More details on how we model the behavior of unsubscribed drivers can be found in Section 3.4. Figure 4.10 shows that DFPS continues to perform better than DFPS-wop, even in the presence of unsubscribed drivers. We also notice that FPA-1 improves the performance for both systems, and DFPS/FPA-1 achieves the lowest

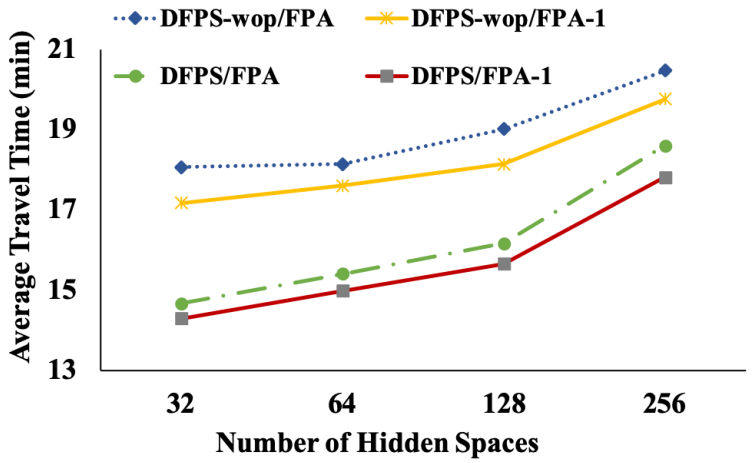

Figure 4.10 Average travel time for different numbers of hidden spaces, 768 drivers, and 8 destinations.

average travel time. These results demonstrate that DFPS/FPA-1 adapts well to the interference caused by unsubscribed drivers.

Individual Travel Time Gains/Losses. In the *subscribed-drivers-only* scenario, we conduct an experiment to find out the gains and losses in travel time for individual drivers when comparing DFPS with a Naive solution, a baseline assignment algorithm that assumes the driver goes to the destination and, after arriving there, she starts a breadth-first-search for parking spaces along the nearby road segments. The Naive solution is similar to what most people do in real life. To measure the gains/losses, we calculate the ratio between the travel time obtained by the Naive solution and the travel time obtained by DFPS for each driver. If the ratio is higher than 1, the driver has benefited from DFPS. Otherwise, the driver has not.

Figure 4.11 plots the distribution of individual travel time gains/losses for all drivers in the experiment. We observe that DFPS manages to improve the travel time for a large majority of drivers (over 95%). Many drivers reduce their travel times by more than an order of magnitude. These results are possible due to the high parking

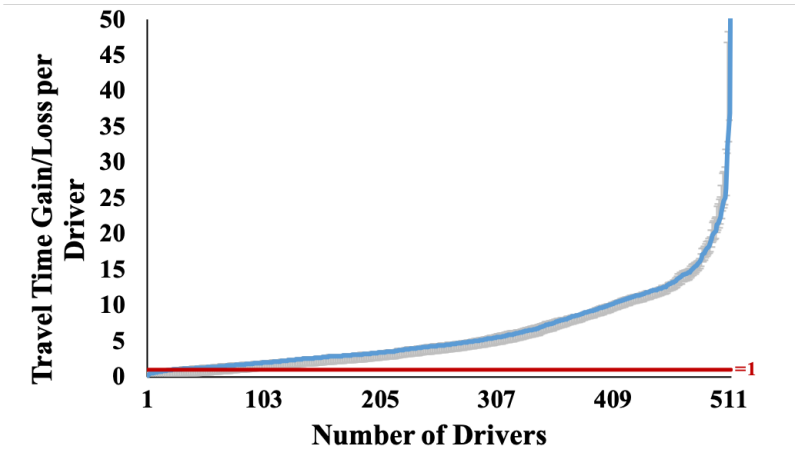

Figure 4.11 Distribution of travel time gain/loss for 512 drivers and 8 destinations. Error bars are shown.

contention generated in the experiment, which leads to high traffic congestion and thus to very high travel times for the Naive solution. The small error bars in the figure demonstrate that these results are consistent across different simulation runs.

Entropy-based cloaking. To determine how our entropy-based cloaking technique affects travel times, we compare its performance with that of a simple k-anonymity technique, which creates cloaking areas containing the $k - 1$ nearest neighbor destinations around the real destination. Figure 4.12 shows the average travel time for when DFPS works with either of these two methods in three cases: DFPS with subscribed-drivers-only, DFPS/FPA with unsubscribed-drivers-interference, and DFPS/FPA1 with unsubscribed-drivers-interference. The results show that DFPS with the entropy-based cloaking technique achieves better performance consistently for all three cases. Therefore, we conclude that the entropy-based cloaking improves both the destination privacy and the travel time. This is because its cloaked region is larger, with destinations spaced more evenly, and thus avoids parking contention and traffic congestion.

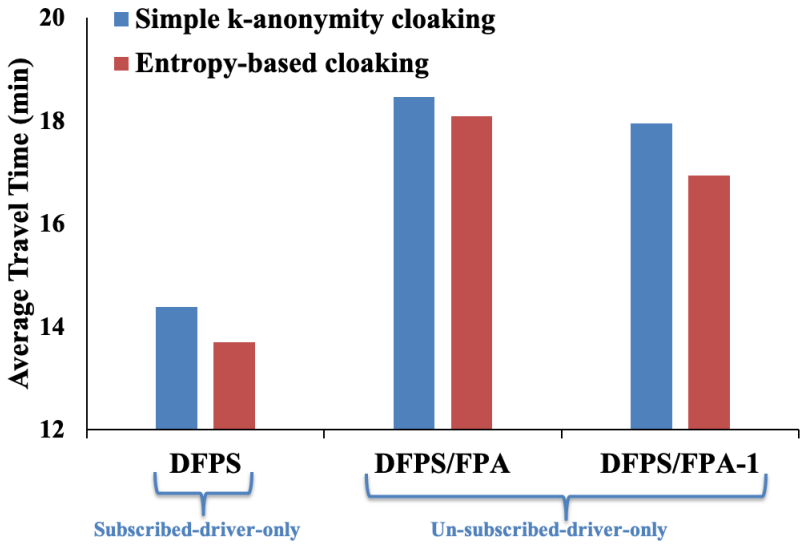

Figure 4.12 Average travel time for simple k-anonymity cloaking vs. entropy-based cloaking for 768 drivers, 8 destinations, and 265 hidden spaces

Effect of region size. To understand how the region size affects performance, we compare the average travel times for DFPS and two versions of DFPS-wop. The minimum number of available parking spaces in a region, P_{min}, is set to 3 for DFPS and one version of DFPS-wop, denoted DFPS-wop(3), and to 6 for the other version of DFPS-wop, denoted DFPS-wop(6).

Table 4.3 shows that the average travel time gradually decrease with larger region sizes. The cloaked regions of DFPS with two and four destinations are smaller than the regions in DFPS-wop, due to two reasons: (1) There are many neighbour destinations around the 2 or 4 destinations chosen in the experiments; this helps reducing cloaked region sizes in DFPS. (2) The parking availability around the destinations is high when drivers submit their parking requests. However, we noticed that the region with eight destinations is larger in DFPS than the regions in DFPS-wop. The reason is that the distance between the 4 new destinations (in addition to the first 4) and their nearest neighbours are relatively large (*i.e.*,

Table 4.3 Average Travel Time for Different Region Sizes, Different Numbers of Destinations, and 768 Drivers

Destinations	2			4			8		
	DFPS	*DFPS-wop(3)*	*DFPS-wop(6)*	*DFPS*	*DFPS-wop(3)*	*DFPS-wop(6)*	*DFPS*	*DFPS-wop(3)*	*DFPS-wop(6)*
Region Size	319.6	356.2	405.8	309.3	366.8	403.6	684.2	371.5	394
Avg. Travel Time	18.1	17.8	16.5	17.9	16.9	16	13.5	15.8	15

sparsely populated region). Thus, to construct a region that satisfies DFPS privacy requirement, the region has to be expanded.

The results show that a slight increase in the region size can improve significantly the travel time in DFPS. This indicates that using larger k values is a good solution: it expected to increase the privacy protection and improve the travel time, at the same time. However, if the regions become too large, it is possible that the parking assignment is done too early and unsubscribed drivers may take some of the assigned spaces. Next, we investigate this trade-off between privacy protection as measured by the value of k and the average travel time.

Impact of increasing the privacy level on average travel time. Figure 4.13 shows how the average travel time and the region size vary with k in an unsubscribed-driver-interference scenario. A number of 256 hidden spaces are taken by unsubscribed drivers gradually at a rate of two spaces every minute. We observe that increasing k leads to larger cloaked regions, which provide better privacy protection. The travel time, however, is not proportional with the region size. It gradually reduces until $k = 7$, and then increases. The explanation for the increase is that parking assignment is done too early for larger regions, and unsubscribed drivers have time to take some of the assigned spaces. The slight decrease in the travel time for $k = 11$ vs. $k = 9$ can be explained by the fact that parking spaces are taken unevenly by unsubscribed drivers. For example, for $k = 11$, the unsubscribed drivers tend to occupy spaces farther away from the real destinations. Overall, the results demonstrate that a good

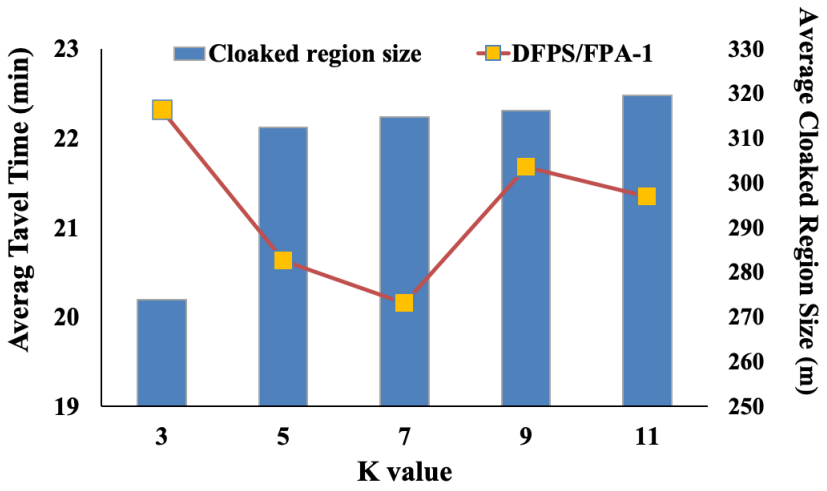

Figure 4.13 Average travel time and cloaked region size for different values of k(-anonymity), 768 drivers, 2 destinations, and 265 hidden spaces.

balance can be found between the level of privacy protection and the travel time (*e.g.,* $k = 7$ in this experiment).

4.7 Summary

This chapter presented DFPS, a cost-effective and efficient distributed mobile system for parking assignment that can be implemented and deployed in real-life settings. DFPS uses the smart phones of the drivers to offload the computation of parking request assignments from a central server, and thus the assignment process becomes scalable in real-time. Parked drivers cooperate to serve parking requests in a distributed fashion while optimizing the social welfare of the whole system, *i.e.,* minimizing the total travel time. DFPS protects the privacy of the drivers' destinations through a novel entropy-based cloaking technique, which guarantees $k-$anonymity. The simulation results demonstrated that DFPS is scalable, effectively reduces the average travel time, and achieves better performance than a centralized system. Furthermore, the results show that achieving destination privacy does not hurt the travel time performance.

CHAPTER 5

MULTI-DESTINATION VEHICULAR ROUTE PLANNING WITH PARKING AND TRAFFIC CONSTRAINTS

This chapter introduces a new instance of multi-destination vehicular route planning problem, which considers real-time parking and traffic constraints, and it presents a solution for this problem, namely the MDVRP system. Section 5.1 presents an overview of the MDVRP system. Section 5.2 defines the optimization criteria for time-dependent route planning and free parking assignment, and it describes the route planning algorithm, TDTSP-FPA. Section 5.3 presents our novel experimental platform and the experimental results obtained on top of this platform. Section 5.4 gives a summary of the chapter.

5.1 System Overview

The goal of MDVRP is to plan a multi-destination route that satisfies real-time traffic and parking conditions, in which the total travel time (*i.e.,* driving and walking times) for all drivers in the system is reduced.

To illustrate how MDVRP works, Figure 5.1 shows the system design of MDVRP system, which consists of two components, namely *Driver Manager* (DM) and *Route Planning Manager* (RPM).

DM is a mobile app that runs on each driver's smart phone, which consists of three modules: *driver request initiator*, *tracker*, and *driver guidance*. DM is in charge of submitting a multi-destination route request and reporting parking status to the RPM. Once it receives a route from RPM, it guides the driver in their trip. The reporting of parking status relies on the app, which can learn this status from an activity recognition service running on the phone [59].

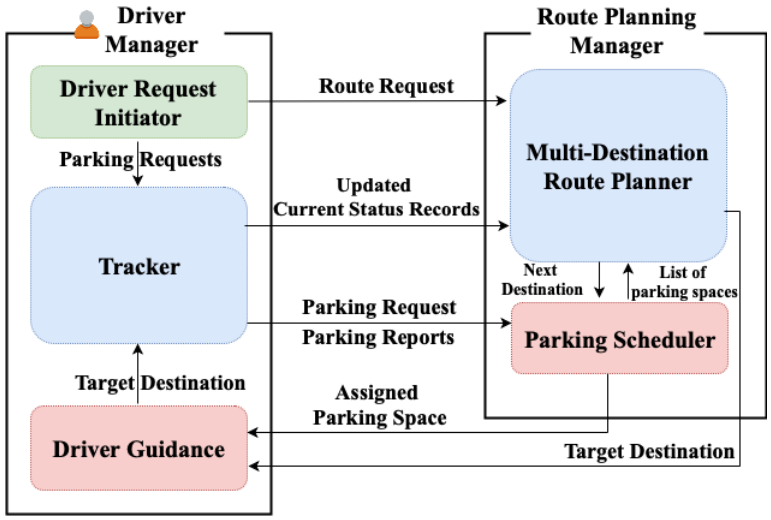

Figure 5.1 MDVRP system overview.

RPM runs on a central server and consists of two modules, the *multi-destination route planner* and the *parking scheduler*. RPM manages the incoming route requests, aggregates the DM parking reports to determine the available parking spaces, and provides multi-destination route planning services to drivers. The services are invoked upon the initial request for trip planning from a driver, and are re-invoked at each destination to plan the remaining route for the driver based on her current location. The TDTSP-FPA algorithm running at RPM combines a solution for the Time-Dependent Traveling Salesman Problem (TDTSP) [19] to find the fastest route for the next destination with our Free Parking Assignment Algorithm (FPA) (see Section 3.4) to find free curbside parking that minimizes the driving plus walking time for all drivers in the system. MDVRP is designed to first consider traffic conditions, and then consider the parking conditions, as drivers approach their destinations.

We now describe the lifecycle of a multi-destination route request in MDVRP, from generation to completion. When a driver submits a request, the *driver request initiator* on her phone generates two types of requests: a route request and several parking requests (corresponding to the multiple destinations). The route request

contains the destinations chosen by the driver and the driver's current-status record, (*i.e.*, driver's current road segment, position on road segment, observation time). The route request is sent to RPM, where all incoming route requests are streamed into a queue by the *multi-destination route planner* module and are processed on a first-come-first-serve basis. The parking requests are forwarded to the *tracker* at the DM, which sends them individually to the *parking scheduler* at the RPM each time the driver approaches a new destination and needs a parking space near that destination. The number of parking requests equals the number of the driver-specified destinations. Each parking request contains a driver-specified destination and the amount of time the driver wants to spend at the destination (*i.e.*, parking duration).

The *multi-destination route planner* manages and serves incoming route requests. It plans routes in a way that optimizes the total travel time. Specifically, for each route request, it uses a time-dependent graph representation of the road network and applies a Time-Dependent Traveling Salesman Problem (TDTSP) solution to compute the fastest path between two given locations. The travel time over a road segment depends on its traffic congestion status, which in turn depends on the time instant at which the road segment is traversed. Thus, knowledge about real-time traffic information over the road network is required. Even though speed profiles extracted from history data can provide a good estimation of long-term traffic dynamics, the short and mid-term forecast of travel times on road segments, particularly the time instant at which the segments are traversed must be made dynamically. Thus, we obtain the time cost of a road segment from existing open source historic trajectory data [74] and real-time traffic information from drivers who are part of our system (*i.e.*, MDVRP drivers) [75]. As shown in [76], drivers' smart phones can form a traffic sensing infrastructure, and a 2-3% penetration of smart phones in the driver population is enough to provide accurate measurements of the velocity of the traffic flow.

The initial routes determined by TDTSP are adjusted after visiting each destination based on the locations of available parking spaces around the next destination. This is done to minimize the total cost of traversing the route, which includes the time spent on both driving to parking spaces and walking to destinations from parking spaces. Since parking spaces may be taken without notice by drivers who are not part of MDVRP, we consider the k closest parking spaces to each destination when computing the routes. To select the next destination, the *multi-destination route planner* averages the travel times between driver's origin location and the k selected parking spaces around each destination. It then selects the destination with the shortest average travel time. Once the next destination is computed, the corresponding route and the destination are sent to the DM's *driver guidance* module.

Given the driver's next target destination, the *driver guidance* module guides the driver to the destination. It also forwards the destination to the *tracker*, which then submits a parking request to the *parking scheduler* when the driver approaches the target destination. If the parking request is sent when the driver is far away from the destination, drivers who are not part of our system (*i.e.*, unsubscribed drivers) have a high likelihood of taking the assigned space. If the request is sent when the driver is too close to the destination, the system may not be able to find a parking space close enough to the destination.

Therefore, as the driver approaches the target destination, we use the Request Distance (see Figure 3.1) to determine when the driver's parking request has to be sent by the *tracker* to the *parking scheduler* in order to be assigned a parking space. As we explained earlier in Section 3.1, this distance defines a circle centered around the destination and its radius was determined experimentally to be initially set to the average length of the roads within the whole region (*i.e.,* zip code). The radius can be adjusted periodically based on the parking occupancy rate in the area which is learned from the RPM (*i.e.,* the radius is increased when the occupancy becomes

higher). RPM may over-estimate the number of available parking spaces as it uses only information from MDVRP drivers. This is because unsubscribed drivers may take parking spaces presumed to be available by our system. This problem is discussed and solved in Section 3.4 based on keeping track of spaces occupied by unsubscribed drivers and on avoiding assigning these spaces for a period of time.

After receiving the parking request, the *parking scheduler* enqueues it for parking scheduling and assignment. The parking assignment decision is made once the Request Distance is reached in such a way as to minimize the total travel time (driving and walking times) of all drivers in MDVRP. The parking assignment algorithm is described in Section 5.2. Once the driver parks in the assigned space, the *parking scheduler* deletes the parking request from the queue. The *tracker* reports the status of the parking space to the *parking scheduler* when the driver is going to either park at or leave the assigned space. When the driver leaves the space, the *tracker* also updates the driver's current-status record and sends it to the *multi-destination route planner* to find the fastest path toward the next target destination in the trip. The aforementioned process is repeated until all the driver-specified destinations are visited.

Both the *multi-destination route planner* and the *parking scheduler* aim to minimize the total travel time; however, the *multi-destination route planner* minimizes the travel time toward the next destination (up to the Request Distance) for each driver. Then, once the Request Distance is reached, the *parking scheduler* minimizes the total travel time (driving from the Request Distance to the parking space and walking from the parking space to the destination and back) for all the drivers.

The design of our MDVRP system is modular and, thus, other time-dependent route planning and parking assignment algorithms can be used. Even though we use the TDTSP's point-to-point shortest path algorithm [19] and the Free Parking

Assignment algorithm (FPA) (Section 3.4), they can be replaced with other such algorithms.

5.2 Travel Time Optimization

We consider the multi-destination route planning problem with parking and traffic constraints defined as follows. Given a sequence of route requests ordered by generation time, we aim to serve each request by finding the fastest route leading drivers to their destinations while considering the real-time traffic and providing free parking assignment service at each destination in the route.

The salient character of our problem lies in that we aim to reduce the total travel time of all drivers as much as it is practically possible. The travel time for each driver is split into: 1) The driving time from the current location to the parking's Request Distance of the next target destination; 2) The driving time from the moment the driver reaches the Request Distance to the moment it parks; 3) The walking time between the parking space and destination (forth and back).

To achieve this goal, we develop the TDTSP-FPA algorithm. TDTSP-FPA uses a solution to TDTSP to solve a multi-destination route planning problem in such a way as to minimize the travel time toward destinations (point 1 above). FPA solves drivers' contention for the same parking spaces in such a way as to optimize the total travel time to each destination in their trips (points 2 and 3 above). TDTSP helps FPA in the sense that it finds the fastest route that avoids traffic congestion to the destination, which implicitly means it is easier to find a parking space along the path. FPA helps TDTSP by reducing the traffic congestion due to cruising while looking for parking.

5.2.1 Optimization Formulation

The optimization objective for our problem is to reduce the total travel time for all drivers. Specifically, the problem targets a set of requesting drivers $V = \{v_1, v_2, ..., v_n\}$; and each driver v_i has a set of target destinations $D = \{d_1, ..., d_z\}$. When planning routes, we also consider curbside parking spaces, which are denoted by $S = \{s_1, s_2, ..., s_m\}$, and the parking occupancy periods for each destination, which are described by w_{s_j}, *i.e.,* the time duration that parking space s_j will be occupied by a driver and cannot be utilized for any other driver.

The drivers are assumed to be moving independently based on legal speeds and on the congestion levels on different road segments. All the geographical locations, including the addresses of destinations and the locations of parking spaces, are converted into latitude and longitude coordinates in the system.

The optimization solves two problems together, TDTSP and FPA, which are as described as follows.

TDTSP Definition TDTSP is a well-known route planning problem for multiple destinations. TDTSP extends the original Traveling Salesman Problem (TSP) with the specific goal of finding the fastest connection on time-dependent road networks. The travel time on the road networks depends on the traffic congestion. All drivers travel along a road network that is modeled as a directed graph $G(N, E)$. Each directed edge $e \in E$ represents a road segment and each node $n \in N$ represents the intersection of two or more roads. Given a segment e_i, it takes time t_i for a driver to travel from one intersection to another along e_i. Note that traffic conditions represented by t_i can be incorporated in the model by introducing weights on graph edges [75]. If a trip begins or ends in the middle of a road segment, we approximate the location to the nearest intersection node. This approximation works well in our city settings, where the road segments are a mix of medium-length and short.

Next, we formally define the concept of path, travel time function, timed-path in a graph, travel time of sub-tour, and we then give an alternative formal definition of the TDTSP.

Definition 1 (Path). *A path* $P = (n_1, ..., n_k)$ *in a graph* $G = (N, E)$ *is a sequence of nodes such that* $(n_i, n_{i+1}) \in E, \quad \forall i \in \{1, ...k-1\}, k \geq 2.$

Definition 2 (Travel Time Function). *A travel time function* $f : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$ *is a function such that for a given edge* $(n_i, n_j) \in E, f(n_i, n_j, t)$ *is the travel time from* n_i *to* n_j *when leaving* n_i *at time* t.

The travel time function dynamically associates travel times to road segments at the time when the segment is traversed, *i.e.*. MDVRP does this based on historical speed profiles as well as frequent updates received from drivers in the system.

Definition 3 (Timed-Path). *Given a graph G=(N,E), a path starting time* $\tau \in \mathbb{R}^+$ *and a travel time function* $f : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, *a timed-path* $P_{\tau,f}$ *in G is a path* $(n_i, ..., n_k)$, *in which each node* n_i *has an associated start time* $t(n_i, P_{\tau,f})$ *such that:*

$$t(n_i, P_{\tau,f}) \geq \tau, \quad \forall i \in \{1, ..., k\}$$

$$t(n_{i+1}, P_{\tau,f}) \geq t(n_i, P_{\tau,f}) + f(n_i, n_{i+1}, t(n_i, P_{\tau,f}))$$

Next, we define the travel time to parking, which is the time between the current location of the driver (origin or current parking space) and its next parking space (*i.e.,* for the next destination). Recall that we do not know which parking space will be available when the car approaches the next destination, and thus consider the k closest parking spaces to the destination in our system.

Definition 4 (Travel Time to Parking). *Given a graph G=(N,E), two nodes* (n_i, n_j) *that represent a driver's current location* (n_i) *and the next target destination* (n_j) *in a driver's route, a current time* t, *and a travel time function* $f : E \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$, *a travel time to parking* T_{ij} *is the average of the minimum costs (i.e., time) timed-paths between the origin* n_i *and the* k *available parking spaces closest to the destination* n_j.

MDVRP calculates the k available parking spaces at the time the vehicle is ready to drive toward the next destination (*i.e.,* MDVRP does not predict the parking availability at the time the vehicle arrives at destination). The parking spaces are calculated based on the occupancy period w_{s_j} and the travel time to the parking space s_j from the current location n_i. If by the time the driver approaches the destination, some of the k parking spaces become unavailable, MDVRP is able to adapt and find other parking spaces.

Definition 5 (TDTSP). *Given a graph G=(N,E), a path starting time $\tau \in \mathbb{R}^+$, a travel time function $f : E \times \mathbb{R}^+ \to \mathbb{R}^+$, and a timed-path $P_{\tau,f}$, TDTSP finds the fastest route which starts from the origin $(n_1 = o)$ and visits each destination exactly once. The route is computed using the travel times to parking, T_{ij}, computed between each pair of (n_i, n_j) nodes.*

FPA Definition As explained earlier in Section 3.4, a parking assignment of spaces to drivers is defined as Y: $V \to S$, where y_{ij} is the assignment of a driver $v_i \in V$ to a parking space $s_j \in S$. For a set of drivers and a set of parking spaces, there may exist a large number of assignments. The algorithm seeks to find an assignment that can minimize the total travel time (driving and walking) of the drivers to each destination in their trips. The travel time $T(v_i)$ toward one destination in a driver v_i's trip is calculated in real-time and consists of two parts, the driving time and the walking time:

- T_d $(O_{v_i},\ s_j)$ is the driving time of driver v_i from the moment she submits her request from location O_{v_i} until she parks at the parking space s_j.

- T_w $(s_j,\ d_{v_i} + s_j)$ is the walking time of the driver between the parking space s_j and the destination d_{v_i} (forth and back).

5.2.2 A Solution for the TDTSP

In the RPM component of our system, we deploy the time-dependent point-to-point shortest path solution [19] to compute a timed-path with minimum travel time to the next destination. This is a bidirectional search algorithm on time-dependent road networks, based on the A* algorithm. The given network is modeled as a directed graph with time-dependent travel time functions for all edges. The algorithm procedure leverages a modified generalization of Dijkstra's algorithm, made bidirectional and improved in several aspects. As for the backward search in A*, the arrival times are not known in advance. Thus, the reversed graph has to be weighted by a lower bound cost (constant travel time for all time instants *i.e.,* edge length/maximum speed limit).

Given a graph G= (N ,E) and origin and destination nodes o, d $\in$ N, the algorithm for computing the fastest o-d path works in three phases.

1. A bidirectional A* search occurs on G, where the forward search is run on the graph weighted by the travel time function, and the backward search is run on the graph weighted by the lower bound cost. All nodes settled by the backward search are included in a set M. Phase one terminates as soon as the two search scopes meet.

2. Suppose that node $n \in N$ is the first node in the intersection of the forward and backward searches, where a time cost of the path going from o to d passing v is an upper bound cost of the path of (o, d, t). In the second phase, both search scopes are allowed to proceed until the backward search queue contains only nodes associated with costs less than the upper bound. Again, all nodes settled by the backward search are added to M.

Algorithm 4 TDTSP-FPA Pseudo-code Executed for Each Visited Destination

Phase one
Input: a driver's origin o_v, set of target destinations $D_v = d_1, ..., d_z$, a value k for the closest parking spaces to each destination, and a starting time τ
$curr\_orig \longleftarrow \mathrm{o}_v$ // current origin of the trip
$rem\_D_v \longleftarrow \mathrm{D}_v$ //set of remaining destinations to be visited
for each destination $d_v^i \in D_v$ **do**
    Define a list of k parking spaces $L_{d_v^i}$ which are the closest available spaces to d_v^i at the approximate time of arrival to d_v^i
    $Origin\_set \longleftarrow \mathrm{D}_v\text{-}d_v^i + curr\_orig$
    for each parking space $s_j \in L_{d_v^i}$ **do**
        for each o in $Origin\_set$ **do**
            Compute travel time $\alpha_o^{s_j}$ of the timed-path between o and s_j at time t
        end for
    end for
    for each o in $Origin\_set$ **do**
        Compute the travel time to parking T_i between o and d_v^i by averaging the travel times αs_{j_o} between o and the k parking spaces
    end for
end for
$fastestRoute \longleftarrow TDTSP(\mathrm{D}_v, T)$
Send first destination, d, in $fastestRoute$ to FPA procedure to assign parking space
Phase two //executed once the driver reaches the Request Distance for parking assignment
Input: a driver's current location c_v and the destination d
Create the list of current available parking spaces L_d in the proximity of d
$s_v \longleftarrow FPA(\mathrm{c}_v, d, L_d)$ //assigned parking space for driver v
Guide v to s_v.
$rem\_D_v \longleftarrow rem\_D_v - d_v^i$
$curr\_orig \longleftarrow s_v$

3. Only the forward search continues, with the additional constraint that only nodes in M can be explored. The forward search terminates when d is settled.

5.2.3 The FPA Algorithm

The *parking scheduler* component runs the FPA algorithm periodically to assign parking spaces to outstanding parking requests in the queue. We determined

experimentally, based on simulations, that running FPA every 2 seconds provides a good trade-off between performance and overhead. In each period, FPA first pre-allocates to the driver of each outstanding request an available parking space that is closest to her destination. The pre-allocation adapts the solution to the flow problem described in the Parking Slot Assignment Games (Psag) [4] to minimize the total walking time of these drivers. The actual assignment of parking spaces takes place based on the urgency of the demands for parking spaces, which is measured by how close the corresponding drivers are to their destinations or their pre-allocated parking spaces. Specifically, in each period, the drivers with the most urgent demand (*i.e.,* they may pass their destination if a parking assignment is not made quickly) are selected and their pre-allocated parking spaces are officially allocated to them. For more details, we direct the reader to a description of the FPA algorithm in Section 3.4.

5.2.4 The TDTSP-FPA Algorithm

The procedure of serving drivers' request in TDTSP-FPA algorithm is divided into two phases, as shown in Algorithm 4, and each phase requires a list of parking spaces that are located in a destination's region. These lists are static, as defined by the municipality data on streets with free curbside parking. Therefore, for each destination, we define an ascending list of parking spaces offline where each parking space is ordered according to the road distance to its associated destination.

The first phase invokes the TDTSP procedure to find the shortest route that starts from a driver's current origin and visits all the destinations once in such a way as to minimize the total travel time. We compute the travel time to parking according to Definition 4 (lines 5-16 in Algorithm 4) for each pair of nodes in the graph (*i.e.,* the union of current origin and the set of remaining destinations not visited yet). Then, we apply TDTSP according to Definition 5 (line 17), and select the first destination in the fastest route generated by TDTSP (line 18). This will be the next destination,

for which FPA will assign parking. In order to reduce the time spent on computing paths, we re-use the paths that have been computed in the past x minutes for drivers who share the same locations and destinations, where x is determined experimentally. In this second phase, the FPA procedure is invoked when a driver reaches the Request Distance (line 22). Once a parking space is assigned, the driver's phone will guide the driver toward this space (line 23). Lines 24-25 update the set of visited destinations and sets the new current origin of the driver. The whole algorithm is executed again to determine the next destination after the parking duration at the current destination expires.

5.3 Experimental Evaluation

We have evaluated the performance of MDVRP and TDTSP-FPA algorithm using simulation with real traffic traces in a real-world road network, which provide us with realistic constraints in terms of traffic and parking.

5.3.1 Evaluation Goals

Our evaluation aims to determine:

- The overall effectiveness of the TDTSP-FPA algorithm on reducing *the average travel time*. The travel time of a driver includes the time spent on driving to the assigned parking locations and the time spent on walking from the parking locations to destinations and then back to the parking locations. It does not include parking duration. The travel times of all drivers in each experiment are averaged to reflect the overall performance.

- Contributions of *driving time* and *walking time* in the total reduction of travel time.

- The *scalability of TDTSP-FPA*, as the percentage of the MDVRP's drivers among all drivers on the roads increases.

- The effectiveness of MDVRP on reducing the travel times of individual drivers. We want to know how many drivers use less time to finish their trips and how many drivers spend more time when MDVRP is used. We calculate the *improvement rate*, which is the proportion of drivers with travel time reduced by MDVRP, to reflect its effectiveness.

- The robustness of the system under a varying compliance rates (*i.e.,* percentage of drivers who follow the suggested visiting order).

5.3.2 Comparison Algorithms

- *Highest Transition Probability Order (HTPO)* represents human mobility habits without careful route planing: a driver always picks the destination that is closest to her current location as her next stop.

- *Traveling Salesman Problem (TSP)* is a classical routing strategy that aims to minimize the total travel distance; it does not consider any constraints. The problem is NP-hard, but a heuristic algorithm for solving the TSP problem is used in the experiments [77].

- *Time-dependent Traveling Salesman Problem (TDTSP)* uses travel time as a metric to select the shortest path between driver's origin and destination that yields the provably fastest route. Paths can be evaluated by considering simply point-to-point shortest paths [19] and real-time traffic density on the road segments [75].

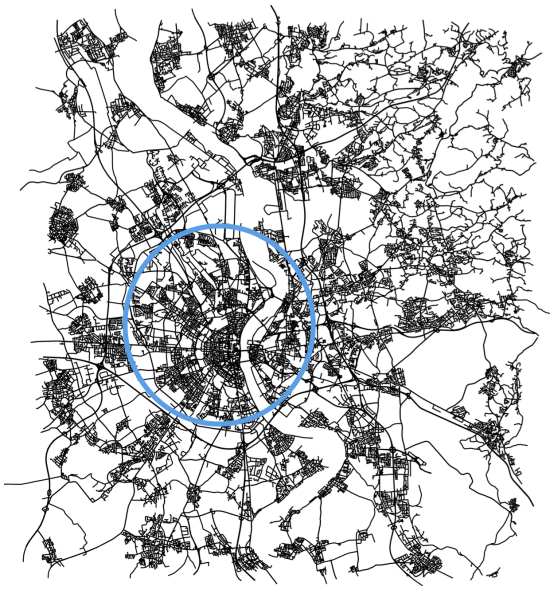

Figure 5.2 Illustration of road network in Cologne.

In HTPO, TSP, and TDTSP, a driver searches for the closest free parking spaces using breadth-first search.

5.3.3 Experimental Platform

Real-World Traffic and Road Network Dataset We use the TAPAS Cologne driver trace [74], which contains the traffic records of over two million drivers in the city of Cologne, Germany during a period of tow hours from 6:00 am to 8:00 am. Each trip record includes a departure time, an origin location and a destination (the IDs of the corresponding road segments), and the route from the origin to the destination. We map the trips to the road network in the same city, which contains 31,584 road intersections and 71,368 road segments. The map is shown in Figure 5.2.

Request Generation The requests used to drive the simulation are derived from the trip records in the TAPAS Cologne dataset. This process allows us to 1) control the number of drivers in simulation experiments; 2) select only the destinations in Cologne downtown (*i.e.,* the centroid area in Figure 5.2) which is the most congested

area in the city, since we are most interested to evaluate TDTSP-FPA in crowded areas with enough vehicular traffic and contention for parking; and 3) have requests with multiple destinations in simulation experiments.

To generate realistic route requests with specific departure times and multiple destinations, we use the method proposed in [78]. We first divide the trips in the dataset into short-time bins, denoted by b_i and denote all road segments by r_i. Then all trips are assigned into bins based on the departure time of the trips. We assume that the destinations of trips on each road segment approximately follow a Poisson distribution during time frame f_j, where each frame has a fixed length spanning L time bins. Thus, the Poisson distribution parameter λ_{ij} is computed for each road segment r_i during time frame f_i. Specifically, for each road segment r_i, we count the number of trips that originated from r_i within time frame f_i, denoted by c_{ij}, and learn the probability distribution of the destination road segments of these trips, denoted by p_{ij}. Then, we calculate λ_{ij} based on c_{ij} using Equation (5.1) and generate a target route request that follows a Poisson process.

$$\lambda_{ij} = c_{ij}/L. \tag{5.1}$$

For each route request generated in frame f_i with the origin road segment r_i, a destination road segment is generated according to the probability distribution p_{ij}. We only consider the destination road segments with high probability distribution in the Cologne downtown area to ensure enough vehicular traffic and enough contention for parking spaces. Note that the dataset only reveals one destination in a trip; however, in reality there are more destinations. To keep the characteristics of a realistic scenario, we repeat the operation and select from the list more trip records with the origin of each trip record being the destination of the previous trip record.

Since trip records are selected according to the probability distribution, they reflect the real distribution of trip destinations in Cologne downtown and the mobility patterns of the drivers. Also note that the drivers that submit requests are not the only drivers in the road network in the simulation, since background traffic is also included in the simulation, as we will discuss in this section. The route requests contain only the trips that we are interested to evaluate.

The route requests have different numbers of destinations (*e.g.,* 1∼7). We set 40% of the routing requests to the largest number of destinations to induce more traffic congestion and to resemble the case of delivery drivers. The rest of the requests are set with fewer destinations to resemble individual drivers. For example, in an experiment with 1∼4 destinations, 40% of requests are set with 4 destinations, 30% with 3 destinations, 20% with 2 destinations, and 10% with 1 destination. To obtain a diverse workload, different simulations have different upper limits.

The length of parking duration is randomly chosen within [10 min, 25 min], to keep the duration reasonable. Note, the time needed to walk from the parking location to the destination and back to the parking location is not included in the parking duration, as it is an important factor in our optimization objective.

We set the value of k, the number of closest available parking spaces to each destination considered in TDTSP, to 3. We found that a small value of k is sufficient to deal with the problem of parking spaces taken by cars that are not part of MDVRP, while avoiding an increase in the computation time. Furthermore, k cannot be very large in order to ensure that the parking spaces are close to the destinations.

Simulation Setup We use SUMO [64] to run vehicular traffic simulations, and use TraaS [79] to send commands to drivers and direct them in their routes. We use the NetEdit tool in SUMO to create travel destinations and parking spaces on the Cologne map. The total number of parking spaces around the destinations is 2400.

To simulate the scenarios with real traffic conditions, we varied the background traffic by including different numbers of additional drivers (40k∼80k). These drivers make single-destination trips, which are randomly selected from the TAPAS Cologne dataset.Background traffic is introduced because we do not assume that all or even a large fraction of drivers will use the MDVRP system. However, we assume that MDVRP drivers are generally representative of the entire driving population.

The background traffic simulates realistic traffic conditions, but it is not used for parking contention for two reasons. First, we selected only a small number of parking spaces for the drivers that we control; there are many more parking spaces that could be used by drivers in the background traffic. Second, we are not interested to evaluate the effect of unsubscribed drivers (*i.e.,* drivers not subscribed to MDVRP) on parking contention in this section. We proposed a solution to this problem elsewhere (see Section 3.4. All experimental results show averages over five runs.

5.3.4 Experimental Results

Figure 5.3 compares the performance of HTPO, TSP, and TDTSP with TDTSP-FPA with the number of drivers varied from 800 to 2400. The background traffic is generated with 60K drivers. As the figure shows, TDTSP-FPA outperforms the competing solutions consistently, and its performance advantage is more prominent when the number of drivers increases. When the number of drivers is 2400, TDTSP-FPA reduces the average travel time by 34%, 29%, and 26%, respectively, compared to HTPO, TSP, and TDTSP. The results demonstrate the substantial impact MDVRP can have on driving and parking in the cities.

The figure also shows that the average travel time grows quickly for HTPO, TSP, and TDTSP when the number of drivers increases. There are two reasons for this behavior. First, traffic conditions are not considered in HTPO and TSP; thus, they may select congested road segments. The comparison between TDTSP and TSP

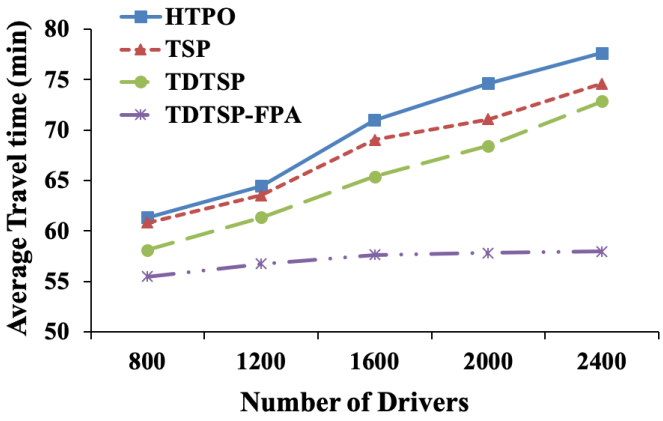

Figure 5.3 Average travel time for a different number of drivers and a varying number of destinations [1∼4].

shows the benefits from taking traffic conditions into consideration. Second, drivers in HTPO, TSP, and TDTSP need to travel more to search for parking, which further increases traffic congestion. TDTSP-FPA directs drivers to parking spaces that are likely to be available. Thus, drivers travel shorter distances looking for parking spaces. This reduces not only their travel time but also the traffic in the road network.

Figure 5.4 breaks down the travel time into two parts: driving time and walking time. The figure shows that drivers spend most time on driving and TDTSP-FPA reduces the average travel times by mostly reducing the driving time. With 2400 drivers, TDTSP-FPA can reduce driving time by up to 54%. Reducing the driving time is very important, as this reduces traffic congestion and implicitly the gas cost and pollution. Since the number of parking spaces in the centroid area is limited, TDTSP-FPA can hardly reduce walking time. We expect that, with the technology developing toward self-driving cars that can drop off drivers at the locations closest to their destinations, the impact of walking time can be ignored in the future. In such a scenario, a self-driving car finds its way to the assigned parking space after dropping off its passenger.

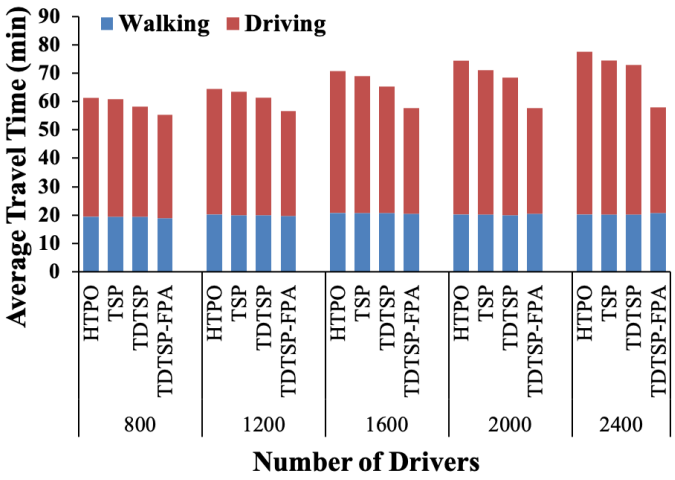

Figure 5.4 Walking and driving time for a different number of drivers and a varying numbers of destinations [1∼4].

The next set of experiments investigate how the travel times change when the number of destinations is varied. Figure 5.5 represents the average travel time for 1200 drivers and 60K background traffic drivers. As the figure shows, TDTSP-FPA reduces the average travel time by larger percentages when the number of destinations increases. For the experiments with 1∼3 destinations, TDTSP-FPA reduces the average travel time by 13% and 7%, respectively, relative to HTPO and TDTSP. For 5∼7 destinations, the percentages increase to 23% and 14% respectively. TDTSP-FPA shows more advantage with more destinations in each trip not only because the traffic in the road network increases, but also because there is more optimization space for TDTSP-FPA to improve parking performance.

We have also investigated how TDTSP-FPA scales when the percentage of MDVRP's drivers increases. To model this scenario, we varied the number of background traffic drivers and kept the number of MDVRP's drivers constant at 2000. The background traffic is generated with 40K, 60K, and 80K drivers. Figure 5.6 shows that TDTSP-FPA decreases the average travel time by 25%, 19%, and 14%, relative to TDTSP, for 40k, 60k, and 80k background drivers, respectively. We observe that

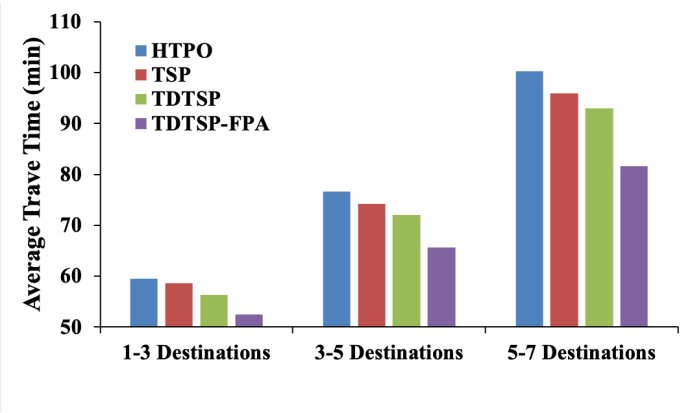

Figure 5.5 Average travel time for a different number of destinations and a fixed number of drivers (1200).

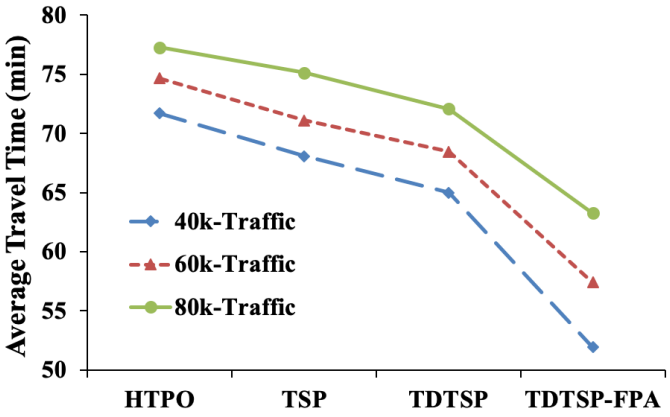

Figure 5.6 Average travel time for 2000 drivers with different patterns of background traffic and a varying number of destinations [1∼4].

TDTSP-FPA scales well, as it reduces the average travel time by larger percentages when the percentage of MDVRP's drivers increases. With more MDVRP drivers, TDTSP-FPA can collect more information from these drivers and affect the traffic more effectively. These results confirm what we observed in Figure 5.3, where we varied the number of MDVRP's drivers, but kept the number of background drivers constant.

While the reduction of average travel time reflects the overall benefits for the drivers in the road network, we also want to find out if most individual drivers spend

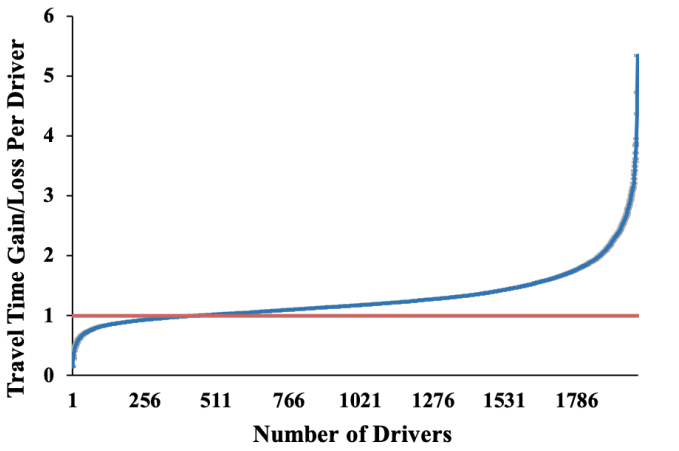

Figure 5.7 Distribution of travel time Gain/Loss for all drivers in the system and a varying number of destinations [1∼4]. Gains are values greater than 1, and Losses are values less than 1.

less time for their trips. Thus, for each driver, we calculate an improvement ratio between the travel time obtained with TSP and the travel time obtained by TDTSP-FPA. A ratio higher than 1 indicates that the driver has benefited from TDTSP-FPA and spent less time with TDTSP-FPA. Then, we sort the drivers based on their ratios, and show the ratios in Figure 5.7. In the experiments, there are 2000 MDVRP drivers with 1∼4 destinations and 60k drivers in background traffic.

As shown in the figure, TDTSP-FPA manages to reduce the travel time for a large majority of drivers (over 85%). However, there are still some drivers who cannot experience improvements. In real-life, these drivers may not know that their time increased, but a few bad experiences could impact the system adoption. Thus, we plan to investigate limiting the number of drivers who experience performance losses and bound performance loss to avoid the worst user experiences.

While it is in the drivers' interest to follow the MDVRP's guidance, it is possible that some drivers will not comply with the guidance (*i.e.,* they will not follow the recommended visiting order of destinations). Therefore, we vary the compliance rate (percentage of drivers who follow the recommended visiting order) to test the system

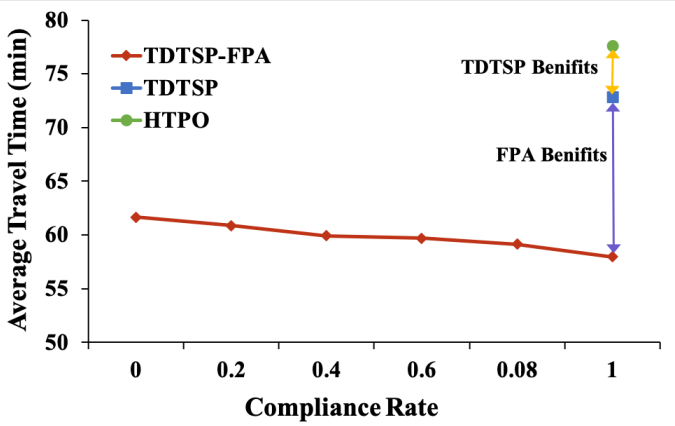

Figure 5.8 Average travel time as a function of the compliance rate for 2400 drivers and a varying number of destinations [1∼4].

robustness. In this experiment, all drivers (including the non-compliant ones) accept the FPA parking assignments. The non-compliant drivers follow their own routes, according to HTPO. Figure 5.8 indicates that MDVRP is robust; compared to TDTSP and HTPO, TDTSP-FPA still offers good improvement, even under a low compliance rate. This is due to the fact that, even at a 0% compliance, drivers still receive benefits from FPA, which in turn can improve the travel time. Conversely, at the higher compliance rate, both FPA and our updated version of TDTSP provide benefits to drivers. The figure shows that the FPA benefits range from 19% to 27%, and the TDTSP benefits are 7% when compared to HTPO.

5.4 Summary

This chapter has addressed a novel problem, namely multi-destination route planning with parking and traffic constraints. This problem has practical applications in many real-life situations, such as package delivery or people visiting multiple destinations in one trip. We formulated this problem analytically in order to optimize the travel time for all drivers. To solve the problem, we designed a novel system, MDVRP, which finds the sequence of destinations that result in the shortest driving and walking time for

the drivers. To the best of our knowledge, this is the first work on multi-destination route planning that considers real-time traffic and parking conditions to optimize the travel time for all drivers in the system. We evaluated the optimization algorithm of MDVRP, namely TDTSP-FPA, over a new and realistic experimental platform that leverages millions of real-life vehicular traces. The experimental results demonstrated that TDTSP-FPA outperforms the comparison baselines, scales well when the number of drivers in MDVRP increases, and is robust to non-compliant drivers. For future work, we plan to optimize the travel time by considering destination arrival deadlines as an additional constraint to our problem.

CHAPTER 6

CONCLUSION

The recent increase in the development and use of smart phones has provided the opportunity to collaboratively sense and share information for the common good. This development has also given rise to solutions that seek to improve the efficiency of the transportation systems. This dissertation proposed cost-effective solutions to tackle the serious mobility problem of drivers who cruise for free vacant parking spaces in urban areas. These solutions are easily deployed in centralized and distributed contexts.

To study the problem in the centralized model, the dissertation proposed FPS, a free parking assignment system, and showed how a centralized server can assign drivers to near optimal parking spaces in order to reduce the total travel time for all drivers. FPS reduces parking space contention because it provides individual space assignment to drivers, which implicitly reduces cruising for parking, traffic congestion, air pollution, and drivers' frustration. The dissertation also presented a novel free parking assignment algorithm for computing this assignment. The performance evaluation of FPS shows that the total travel time for all drivers is reduced even when many parking spaces are occupied by unsubscribed drivers.

To improve scalability and privacy, the dissertation proposed DFPS, a distributed free parking assignment system, which takes advantages of the smart phones of the drivers to cooperatively compute and forward the parking assignments. DFPS uses a central dispatcher to receive and distribute parking requests. The distributed structure of drivers' smart phones, represented as a K-D tree, allows DFPS to increase parallel processing and decrease the response time. DFPS scales well by performing localized computations over smart phones of drivers parked in

proximity of each other. A cloaking-based entropy technique is proposed to preserve drivers' destinations privacy at the dispatcher side, without seeking help from any centralized third party. DFPS deploys the free parking assignment algorithm in a distributed fashion. Extensive experiments show that the distributed solutions (*i.e.,* DFPS and DFPS-wop) can provide better travel time compared to the centralized counterparts, while protecting drivers' destinations.

Finally, the dissertation addressed a novel problem, named multi-destination vehicular route planning with real-time parking and traffic constrains. To solve this problem, a multi-destination vehicular route planning system, MDVRP, was proposed. MDVRP uses a novel algorithm to find a route that visits the destinations in the most efficient order and also assigns free parking spaces to drivers while optimizing a system-wide objective (*i.e.,* total travel time). Through a series of experimental evaluations, we demonstrate that the routing algorithm in MDVRP delivers excellent performance when compared to the baseline algorithms. To the best of our knowledge, this is the first work on route planning that considers handling parking and traffic constraints for multi-destinations as well as optimizing the travel time for all drivers, simultaneously.

The three practical and cost-effective parking assignment systems can be implemented and deployed in real-life settings to manage the parking problems and help in reducing traffic congestion.

REFERENCES

[1] J. P. Rodrigue, C. Comtois, and B. Slack, *The Geography of Transport Systems*, 4th ed. Taylor & Francis Ltd, 2017.

[2] D. C. Shoup, "Cruising for Parking," *Transport Policy*, vol. 13, no. 6, pp. 479–486, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.tranpol.2006.05.005

[3] S. F. M. T. Authority, "Sfpark," https://sfpark.org, San Francisco, CA, USA, [Online; accessed 02-April-2020].

[4] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta, and J. Lin, "Parking Slot Assignment Games," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2011, pp. 299–308.

[5] Y. Xu, E. Alfonsetti, P. C. Weeraddana, and C. Fischione, "A Semi Distributed Approach for the Feasible Min-Max Fair Agent-Assignment Problem With Privacy Guarantees," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 333–344, 2018. [Online]. Available: http://doi.org/10.1109/TCNS.2016.2609151

[6] M. Duckham and L. Kulik, "A Formal Model of Obfuscation and Negotiation for Location Privacy," in *Proceedings of Proceedings of the 3rd International Conference on Pervasive Computing*, Springer. Springer-Verlag, 2005, pp. 152–170.

[7] H. Kido, Y. Yanagisawa, and T. Satoh, "An Anonymous Communication Technique Using Dummies for Location-based Services," in *Proceeding of International Conference on Pervasive Services (ICPS'05)*. IEEE, 2005, pp. 88–97.

[8] B. Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *Proceedings of 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, 2005, pp. 620–629.

[9] F. Dürr, P. Skvortsov, and K. Rothermel, "Position Sharing for Location Privacy in Non-Trusted Systems," in *Proceedings of IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, 2011, pp. 189–196.

[10] S. Lin and B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973. [Online]. Available: http://www.jstor.org/stable/169020

[11] B. Strasser, "Dynamic Time-Dependent Routing in Road Networks Through Sampling," in *Proceedings of 17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'17)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[12] C. Malandraki and M. S. Daskin, "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms," *Transportation science*, vol. 26, no. 3, pp. 185–200, 1992. [Online]. Available: https://doi.org/10.1287/trsc.26.3.185

[13] H. Abeledo, R. Fukasawa, A. Pessoa, and E. Uchoa, "The Time Dependent Traveling Salesman Problem: Polyhedra and Algorithm," *Mathematical Programming Computation*, vol. 5, no. 1, pp. 27–55, 2013. [Online]. Available: https://doi.org/10.1007/s12532-012-0047-y

[14] P. thaisombut, "Generalization of EDF and LLF: Identifying All Optimal Online Algorithms for Minimizing Maximum Lateness," *Algorithmica*, vol. 50, no. 3, p. 312–328, Mar. 2008. [Online]. Available: http://www.cs.pitt.edu/utp

[15] E. Nardelli, "Distributed Kd Trees," in *Proceedings of 16th Conference of Chilean Computer Science Society (SCCC'96)*. Citeseer, 1996, pp. 142–154.

[16] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure Automated Valet Parking: A Privacy-Preserving Reservation Scheme for Autonomous Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 11 169–11 180, 2018. [Online]. Available: http://doi.org/10.1109/TVT.2018.2870167

[17] L. Zhu, M. Li, Z. Zhang, and Z. Qin, "ASAP: An Anonymous Smart-Parking and Payment Scheme in Vehicular Networks," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2018. [Online]. Available: http://doi.org/10.1109/TDSC.2018.2850780

[18] L. Ni, F. Tian, Q. Ni, Y. Yan, and J. Zhang, "An Anonymous Entropy-based Location Privacy Protection Scheme in Mobile Social Networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–19, 2019. [Online]. Available: https://doi.org/10.1186/s13638-019-1406-4

[19] G. Nannicini, "Point-to-point Shortest Paths on Dynamic Time-Dependent Road Networks," *European Alliance for Innovation*, vol. 8, no. 3, pp. 327–330, 2010. [Online]. Available: https://doi.org/10.1007/s10288-010-0121-0

[20] G. Yan, S. Olariu, M. C. Weigle, and M. Abuelela, "SmartParking: A Secure and Intelligent Parking System Using NOTICE," in *Proceedings of 11th IEEE International Conference on Intelligent Transportation Systems*, 2008, pp. 569–574.

[21] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "ParkNet: Drive-by Sensing of Road-side Parking Statistics," in *Proceedings of 8th International Conference on Mobile Systems, Applications, and Services*, 2010, pp. 123–136.

[22] V. Verroios, V. Efstathiou, and A. Delis, "Reaching Available Public Parking Spaces in Urban Environments Using Ad Hoc Networking," in *Proceedings of 12th*

IEEE International Conference on Mobile Data Management, vol. 1, 2011, pp. 141–151.

[23] O. Wolfson, B. Xu, and H. Yin, "Dissemination of Spatial-Temporal Information in Mobile Networks with Hotspots," in *Proceedings of International Workshop on Databases, Information Systems, and Peer-to-Peer Computing*. Springer, 2004, pp. 185–199.

[24] N. BESSGHAIER, M. Zargayouna, and F. Balbo, "An Agent-Based Community to Manage Urban Parking," *Advances in Intelligent and Soft Computing*, vol. 155, pp. pp 17–22, Jan. 2012. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00861808

[25] M. Caliskan, D. Graupner, and M. Mauve, "Decentralized Discovery of Free Parking Places," in *Proceedings of 3rd International Workshop on Vehicular Ad Hoc Networks*, 2006, pp. 30–39.

[26] S. Nawaz, C. Efstratiou, and C. Mascolo, "Parksense: A Smartphone Based Sensing System for On-Street Parking," in *Proceedings of 19th Annual International Conference on Mobile Computing & Networking*, 2013, pp. 75–86.

[27] R. Salpietro, L. Bedogni, M. Di Felice, and L. Bononi, "Park Here! A Smart Parking System Based on Smartphones' Embedded Sensors and Short Range Communication Technologies," in *Proceedings of 2nd IEEE World Forum on Internet of Things (WF-IoT)*, 2015, pp. 18–23.

[28] R. Arnott and J. Rowse, "Downtown Parking in Auto City," *Regional Science and Urban Economics*, vol. 39, no. 1, pp. 1–14, 2009. [Online]. Available: https://doi.org/10.1016/j.regsciurbeco.2008.08.001

[29] D. Mackowski, Y. Bai, and Y. Ouyang, "Parking Space Management via Dynamic Performance-based Pricing," *Transportation Research Part C: Emerging Technologies*, vol. 59, pp. 66 – 91, 2015. [Online]. Available: https://doi.org/10.1016/j.trpro.2015.06.010

[30] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Pricing of Parking for Congestion Reduction," in *Proceedings of 20th International Conference on Advances in Geographic Information Systems*, 2012, pp. 43–51.

[31] H. Wang and W. He, "A Reservation-based Smart Parking System," in *Proceedings of IEEE International Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2011, pp. 690–695.

[32] P. Basu and T. D. Little, "Networked Parking Spaces: Architecture and Applications," in *Proceedings of 56th IEEE International Conference on Vehicular Technology Conference*, vol. 2, 2002, pp. 1153–1157.

[33] S. P. Company, "Spothero," https://spothero.com, Chicago, IL, USA, [Online; accessed 02-April-2020].

[34] A. M. S. C. Company, "Pango," https://www.mypango.com, Fort Lauderdale, FL, USA, [Online; accessed 02-April-2020].

[35] I. Inc, "Parkme," https://www.parkme.com, Santa Monica, CA, USA, [Online; accessed 02-April-2020].

[36] P. Company, "Bestparking," https://www.bestparking.com, Chicago, IL, USA, [Online; accessed 02-April-2020].

[37] T. Delot, N. Cenerario, S. Ilarri, and S. Lecomte, "A Cooperative Reservation Protocol for Parking Spaces in Vehicular Ad Hoc Networks," in *Proceedings of 6th International Conference on Mobile Technology, Application & Systems*, 2009, pp. 1–8.

[38] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Parking in Competitive Settings: A Gravitational Approach," in *Proceedings of 13th IEEE International Conference on Mobile Data Management*, 2012, pp. 27–32.

[39] M. Gruteser and B. Hoh, "On the Anonymity of Periodic Location Samples," in *Proceedings of International Conference on Security in Pervasive Computing*. Springer, 2005, pp. 179–192.

[40] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid," in *Proceedings of 17th International Conference on World Wide Web*, 2008, pp. 237–246.

[41] B. Gedik and L. Liu, "Protecting Location Privacy with Personalized K-anonymity: Architecture and Algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, 2007. [Online]. Available: http://doi.org/10.1109/TMC.2007.1062

[42] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous Location-based Queries in Distributed Mobile Systems," in *Proceedings of 16th International Conference on World Wide Web*, 2007, pp. 371–380.

[43] K. P. S. S. Ghinita, Gabriel, "MOBIHIDE: a Mobilea Peer-to-Peer System for Anonymous Location-based Queries," in *Proceedings of International Symposium on Spatial and Temporal Databases*. Springer, 2007, pp. 221–238.

[44] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A Peer-to-Peer Spatial Cloaking Algorithm for Anonymous Location-based Service," in *Proceedings of 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, 2006, pp. 171–178.

[45] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald, "Privacy-Aware Location Sensor Networks," in *Proceedings of 9th Conference on Hot Topics in Operating Systems*, ser. HOTOS'03. USENIX Association, 2003, p. 28.

[46] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-based Services Through Spatial and Temporal Cloaking," in *Proceedings of 1st International Conference on Mobile Systems, Applications and Services*, 2003, pp. 31–42.

[47] T. Xu and Y. Cai, "Location Anonymity in Continuous Location-based Services," in *Proceedings of 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, 2007, pp. 1–8.

[48] T. Xu and Y. Cai, "Exploring Historical Location Data for Anonymity Preservation in Location-based Services," in *Proceedings of 27th IEEE International Conference on Computer Communications (INFOCOM'08)*, 2008, pp. 547–555.

[49] O. Abul, F. Bonchi, and M. Nanni, "Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases," in *Proceedings of 24th IEEE International Conference on Data Engineering*, 2008, pp. 376–385.

[50] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The New Casper: Query processing for Location Services without Compromising Privacy," in *Proceedings of 32nd International Conference on Very Large Data Bases*, 2006, pp. 763–774.

[51] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Privacy-Preserving Smart Parking Navigation Supporting Efficient Driving Guidance Retrieval," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6504–6517, 2018. [Online]. Available: http://doi.org/10.1109/TVT.2018.2805759

[52] A. Montero, I. Méndez-Díaz, and J. J. Miranda-Bront, "An Integer Programming Approach for the Time-Dependent Traveling Salesman Problem with Time Windows," *Computers & Operations Research*, vol. 88, pp. 280–289, 2017. [Online]. Available: https://doi.org/10.1016/j.cor.2017.06.026

[53] J. Hurkała, "Time-Dependent Traveling Salesman Problem with Multiple Time Windows," *Annals of Computer Science and Information Systems*, vol. 6, pp. 71–78, 2015. [Online]. Available: http://dx.doi.org/10.154392015311

[54] Y. Huang, B.-H. Lin, and V. S. Tseng, "Efficient Multi-Destinations Route Planning with Deadlines and Cost Constraints," in *Proceedings of 18th IEEE International Conference on Mobile Data Management (MDM)*, 2017, pp. 228–233.

[55] P. A. Melgarejo, P. Laborie, and C. Solnon, "A Time-Dependent No-Overlap Constraint: Application to Urban Delivery Problems," in *Proceedings of International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*. Springer, 2015, pp. 1–17.

[56] G. Inc, "Route4me route planner," https://route4me.com, Hackensack, NJ, USA, [Online; accessed 02-April-2020].

[57] Gsmtasks Inc, "Gsmtasks," https://gsmtasks.com, Walnut, CA, USA, [Online; accessed 02-April-2020].

[58] R. B.V., "Routexl: Fastest route with multiple stops," https://routexl.com, Heiloo, Netherlands, [Online; accessed 02-April-2020].

[59] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, and D. J. Cook, "Simple and Complex Activity Recognition Through Smart Phones," in *Proceedings of 8th International Conference on Intelligent Environments*, 2012, pp. 214–221.

[60] M. Arab and T. Nadeem, "Magnopark - locating on-street parking spaces using magnetometer-based pedestrians' smartphones," in *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2017, pp. 1–9.

[61] A. Nandugudi, T. Ki, C. Nuessle, and G. Challen, "PocketParker: Pocketsourcing Parking Lot Availability," in *Proceedings of ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014, pp. 963–973.

[62] B. Xu, O. Wolfson, J. Yang, L. Stenneth, S. Y. Philip, and P. C. Nelson, "Real-time street parking availability estimation," in *Proceedings of 14th IEEE International Conference on Mobile Data Management*, vol. 1, 2013, pp. 16–25.

[63] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*. John Wiley & Sons, Inc., 1998.

[64] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban MObility: An overview," in *Proceedings of 3rd International Conference on Advances in System Simulation (SIMUL)*, 2011, pp. 63–68.

[65] R. W. Bohannon, A. W. Andrews, and M. W. Thomas, "Walking Speed: Reference Values and Correlates for Older Adults," *Journal of Orthopaedic & Sports Physical Therapy*, vol. 24, no. 2, pp. 86–90, 1996. [Online]. Available: https://www.jospt.org/doi/10.2519/jospt.1996.24.2.86

[66] T. Öberg, A. Karsznia, and K. Öberg, "Basic Gait Parameters: Reference Data for Normal Subjects, 10-79 Years of Age," *Journal of Rehabilitation Research and Development*, vol. 30, pp. 210–210, 1993. [Online]. Available: https://doi.org/10.1682/JRRD.2003.07.0361

[67] X. Chen and J. Pang, "Measuring Query Privacy in Location-based Services," in *Proceedings of 2nd ACM conference on Data and Application Security and Privacy*, 2012, pp. 49–60.

[68] C.-Y. Chow and M. F. Mokbel, "Enabling Private Continuous Queries for Revealed User Locations," in *Proceedings of International Symposium on Spatial and Temporal Databases*. Springer, 2007, pp. 258–275.

[69] L. Sweeney, "K-anonymity: A Model for Protecting Privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002. [Online]. Available: https://doi.org/10.1142/S0218488502001648

[70] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-based Identity Inference in Anonymous Spatial Queries," *IEEE transactions on knowledge and data engineering*, vol. 19, no. 12, pp. 1719–1733, 2007. [Online]. Available: https://doi.org/10.1109/TKDE.2007.190662

[71] C. Zhang and Y. Huang, "Cloaking Locations for Anonymous Location Based Services: A Hybrid Approach," *GeoInformatica*, vol. 13, no. 2, pp. 159–182, 2009. [Online]. Available: https://doi.org/10.1007/s10707-008-0047-2

[72] G. Tsatsanifos, D. Sacharidis, and T. Sellis, "Midas: Multi-Attribute Indexing for Distributed Architecture Systems," in *Proceedings of International Symposium on Spatial and Temporal Databases*. Springer, 2011, pp. 168–185.

[73] A. Montresor and M. Jelasity, "PeerSim: A Scalable P2P Simulator," in *Proceedings of 9th IEEE International Conference on Peer-to-Peer Computing*, 2009, pp. 99–100.

[74] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and Analysis of a Large-Scale Urban Vehicular Mobility Dataset," *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 1061–1075, 2013. [Online]. Available: http://doi.org/10.1109/TMC.2013.27

[75] J. S. Pan, I. S. Popa, and C. Borcea, "Divert: A Distributed Vehicular Traffic Re-Routing System for Congestion Avoidance," *IEEE Transactions on Mobile Computing*, vol. 16, no. 1, pp. 58–72, 2016. [Online]. Available: https://hal.inria.fr/hal-01426424

[76] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen, "Evaluation of Traffic Data Obtained via GPS-Enabled Mobile Phones: The Mobile Century Field Experiment," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 4, pp. 568–583, 2010. [Online]. Available: https://doi.org/10.1016/j.trc.2009.10.006

[77] E. L. Lawler, "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization," *Wiley-Interscience Series in Discrete Mathematics*, 1985. [Online]. Available: https://doi.org/10.1112/blms/18.5.514

[78] S. Ma, Y. Zheng, and O. Wolfson, "T-Share: A large-Scale Dynamic Raxi Ridesharing Service," in *Proceedings of 29th IEEE International Conference on Data Engineering (ICDE)*, 2013, pp. 410–421.

[79] S.-S. of Urban Mobility, "Traci/Traas-Sumo," https://sumo.dlr.de/wiki/TraCI/TraaS.html/, Berlin, Germany, [Online; accessed 02-April-2020].