Congestion Mitigation for Planned Special Events:

Parking, Ridesharing and Network Configuration

by

Jun Xiao

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2019 by the
Graduate Supervisory Committee:

Yingyan Lou, Chair
Xuesong Zhou
Ram Pendyala
Pitu Mirchandani

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

This dissertation investigates congestion mitigation during the ingress of a planned special event (PSE). PSEs would impact the regular operation of the transportation system within certain time periods due to increased travel demand or reduced capacities on certain road segments. For individual attendees, cruising for parking during a PSE could be a struggle given the severe congestion and scarcity of parking spaces in the network. With the development of smartphones-based ridesharing services such as Uber/Lyft, more and more attendees are turning to ridesharing rather than driving by themselves. This study explores congestion mitigation during a planned special event considering parking, ridesharing and network configuration from both attendees and planner's perspectives.

Parking availability (occupancy of parking facility) information is the fundamental building block for both travelers and planners to make parking-related decisions. It is highly valued by travelers and is one of the most important inputs to many parking models. This dissertation proposes a model-based practical framework to predict future occupancy from historical occupancy data alone. The framework consists of two modules: estimation of model parameters, and occupancy prediction. At the core of the predictive framework, a queuing model is employed to describe the stochastic occupancy change of a parking facility.

From an attendee's perspective, the probability of finding parking at a particular parking facility is more treasured than occupancy information for parking search. However, it is

hard to estimate parking probabilities even with accurate occupancy data in a dynamic environment. In the second part of this dissertation, taking one step further, the idea of introducing learning algorithms into parking guidance and information systems that employ a central server is investigated, in order to provide estimated optimal parking searching strategies to travelers. With the help of the Markov Decision Process (MDP), the parking searching process on a network with uncertain parking availabilities can be modeled and analyzed.

Finally, from a planner's perspective, a bi-level model is proposed to generate a comprehensive PSE traffic management plan considering parking, ridesharing and route recommendations at the same time. The upper level is an optimization model aiming to minimize total travel time experienced by travelers. In the lower level, a link transmission model incorporating parking and ridesharing is used to evaluate decisions from and provide feedback to the upper level. A congestion relief algorithm is proposed and tested on a real-world network.

DEDICATION

To

My Parents

Jian Xiao and Yanli Qiu

and

My wife

Yue Liu

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to Dr. Yingyan Lou, for providing me the opportunity to pursue my Ph.D. degree in Transportation Engineering and teaching me how to conduct research from scratch. This work would have been impossible without her advice and support. Her patience, motivation and immense knowledge helped me all the way through my research.

I would like to thank my committee members for the substantial influence that their courses have had on my research. Many thanks to Dr. Xuesong Zhou for his great guidance in traffic flow theory and his instruction in operations research while I was taking a reading and conference course with him. Dr. Pitu Mirchandani leaded me to network flow theory and stochastic decision-making area. His insightful comments and hard questions incented me to dive deeper into my research. Dr. Ram Pendyala taught me activity-travel behavior modeling. For the weekly transportation seminars, he invited many great speakers from both industry and research community who widen my sight.

I would also like to thank all my friends and colleagues for occurence during my past five meaningful years. I would especially show my appreciate to Dr. Peiheng Li, Dr. Jiangtao Liu and his family, Dr. Sravani Vadlamani, Josh Frisby for their help on my set up in the U.S. I wounld like to show thanks to Xiushaung Li, Guanqi Fang, Dening, Peng, Di Yang for our happy basketball hours every Saturday. I wounld like to show thanks to Dr. Jianrui Miao, Dr. Monireh Mahmoudi, Hossein Jalali Dr. Yunchao Qu, Dr. Junhua Chen, Jiawei Lu, Xin Wu, Qixiu Cheng for the time worked together in Colleve Avenue

Commons. I wounld like to show thanks to Golnoosh Miri, Yudi, Lei, Taehooie Kim, Shivam Sharda, Denise Capasso da Silva for those times working together on Institution of Transportation Engineers (ITE) events.

Last but not the least, I am deeply grateful to my parents and my wife for their endless love and support.

TABLE OF CONTENTS

Page

APPENDIX Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

1 INTRODUCTION

A planned special event (PSE) such as a sports game or concert is a public activity with a scheduled time and predefined location. PSEs may impact the regular operations of the transportation system within certain time periods due to increased travel demand or reduced capacity. Typically, a PSE network suffers from the increased travel demand during ingress and egress. The egress resembles an evacuation event to some extent, which has drawn tremendous attention from the research community. However, the ingress of a PSE is more challenging and has been a neglected issue. The focus of this dissertation is on congestion mitigation during the ingress of a PSE.

One of the major challenges faced by attendees during ingress is searching for parking spaces, especially for those without reserved parking. The cruising for parking is a waste of time and resources, and could make the transportation network more congested. During a PSE, the road network is usually setup differently from a normal day to channel traffic, including lane reversal, lane closure and turning movement assignment. This makes it harder for travelers with or without reserved parking to get to their destination. This unfamiliarity of the network can be resolved by providing recommended routes and guidance. Ridesharing, as an increasingly important component of the PSE network, also needs to be well planned. Intensive lane-changing behaviors would occur on the ridesharing link, causing severe congestion. Thus, ridesharing drop-off locations should be carefully selected. These problems mentioned above are all closely connected with each other; and their interactions should not be overlooked. A comprehensive approach to

systematically generate a PSE traffic plan considering all the elements: parking, ridesharing, route recommendation and lane configurations is an urgent need.

Attendees could spend significant time cruising for parking due to high parking demand and/or their unfamiliarity to the network. And the cruising, together with enormous inbound traffic would substantially worsen the traffic condition. To help attendees find parking space more efficiently, we propose a model-based practical parking occupancy prediction framework grounded in the underlying stochastic queuing process. This framework is able to forecast parking availability information based on historical data alone. It is the first to validate a model-based parking occupancy prediction framework using real data. We also provide some insights on the applicability and suitability of the proposed model-based framework by comparing it with pure machine learning parking occupancy prediction methods.

However, the effectiveness of providing parking information (current parking availability, predicted future parking availability, and predicted parking search time) alone to users might be limited. Providing travelers with parking searching strategies (suggested actions for a user to take in any given state of the parking searching process) could be a more effective approach to parking management, especially for drivers that are unfamiliar with the network during special events. We propose a modified Q-learning algorithm that can be adopted by such parking guidance and information services to provide user-optimal parking search strategies for individual drivers. It is the first to introduce reinforcement

learning into parking guidance and information systems to provide user-optimal parking strategies to individuals.

Providing parking search strategy could minimize travelers' expected travel time but might not bring benefits to the network from a planner's perspective. With the purpose of minimizing total travel time for the whole network, we propose an optimization model to generate a comprehensive traffic plan for PSEs. Lane reconfigurations (lane reversal, lane closure) are adopted to accommodate the unbalanced travel demand between two directions of a road section during ingress. Other than parking and lane reversal, nowadays ridesharing is also an essential part of PSEs. With the consideration of all these components together, we propose a bi-level model. The upper level is an optimization model aiming to minimize total travel time consumed by travelers. The decision variables are the number of lanes on each link, route recommendations, and ridesharing drop-off locations. In the lower level, a link transmission model (LTM) incorporating parking and ridesharing is used to evaluate the total travel time and provide feedback to the upper level. A heuristic algorithm based on congestion relief is proposed and tested by real data collected from Super Bowl XLIX at Glendale, Arizona.

2 BACKGROUND

A planned special event (PSE) such as a sports game or a concert at the heart of an urban area can greatly affect the regular operation of a transportation system due to significantly increased travel and parking demand that often leads to severe congestion. Previous studies have reported PSEs as a significant contributing factor in causing traffic congestion (Kwon, Mauch, and Varaiya 2007). The vehicles cruising for parking, together with massive unbalanced traffic demand and vehicles providing ridesharing service interact with each other in the network, make PSE transportation planning during the ingress a great challenge. Lane-reversal, parking and ridesharing are some common strategies to mitigate congestion for PSEs. Since these strategies are closely coupled with each other in a PSE transportation network, a well-planned comprehensive transportation plan plays a pivotal role in the success of a PSE.

Searching for parking is a real struggle faced by many drivers, especially in a congested network (IBM 2011). Empirical studies conducted in the US suggested that the average time spent searching for a curbside parking space ranged between 3.5 and 14 minutes (Shoup 2006), which has a tremendous negative impact on a traffic network. According to Shoup (2006), cruising for parking is responsible for 30% of traffic on average in downtown areas. In a more recent research, Giuffrè et al. (2012) found that cruising for parking results in a peak increase of about 25 – 40% of the traffic flow. Moreover, congestion caused by cruising for parking is a waste of resources and aggravates environmental issues. Ayala (2012) concluded that for a city like Chicago, 8.37 million

gallons of gasoline would be consumed and 129 thousand tons of CO_2 emitted every year due to cruising for parking. The significance of the parking searching problem has led to increasing public demand for parking information and services recently. With the proliferation of advanced smartphones, a range of smartphone-based parking management services began to emerge. In fact, the International Parking Institute identified the prevalence of mobile applications the #2 emerging trend in parking in 2015 (IPI, 2015). Available services already launched on the market include mobile payment (e.g., ParkMobile and Pango) as well as parking information provision and reservation (e.g., ParkMe, BestParking, Parkopedia, SpotHero, ParkWhiz). In addition to parking rates and facility properties provided by almost every parking application, ParkMe also offers real-time parking occupancy information (as a percentage) in selected markets. Moreover, some recent empirical studies also found that parking occupancy information is mostly valued by drivers (Caicedo et al., 2006; van der Waerden et al., 2011) and such occupancy information (Caicedo et al., 2006), related variables such as time spent searching for parking (Ibeas et al, 2014) and the availability of parking after a reasonable search time (Chaniotakis and Pel, 2015) do significantly affect drivers' parking choices.

In view of the significance of predictive parking occupancy information in parking searching, Chapter 3 proposes a model-based practical framework to predict future occupancy from historical occupancy data alone.

To further address the parking search problem from an individual event-goer's perspective, Chapter 4 investigates the idea of introducing learning algorithms into

parking guidance and information systems that employ a central server, in order to provide estimated optimal parking searching strategies to travelers. We propose an algorithm based on Q-learning, where the topology of the underlying transportation network is incorporated. This modification allows us to reduce the size of the problem, and thus the amount of data required to learn the optimal strategy dramatically. Numerical experiments conducted on a toy network show that the proposed learning algorithm outperforms the nearest-node greedy search strategy and the original Q-learning algorithm. Sensitivity analysis regarding the desired amount of training data is also performed as backup support.

Other than the parking search, ridesharing is also becoming a critical role during a special event. Ridesharing, which has appeared in different forms since the 1940s, has been considered as a powerful strategy to reduce congestion, emission and travel cost. It evolved from relative long-distance, acquisition-based commute trips to on-demand, smart-phone based trips for various purposes. For a thorough review of the development of ridesharing, please refer to (Chan and Shaheen 2012; Furuhata et al. 2013). In this dissertation, our focus is the app-based on-demand ridesharing service such as Uber and Lyft. These ridesharing services provide a platform dynamically matching drivers with passengers based on their itineraries. Riders submit their trip requests and pay through a smartphone and drivers would be assigned to complete trips for money. In recent years, the proliferation of ridesharing services such as Uber/Lyft made it possible for event-goers to avoid the effort of driving and searching for parking. Special event planners are

encouraging and adapting to the increasing ridesharing demand by designating ridesharing drop-off locations, ridesharing-only lanes or areas. Uber and Lyft have also launched their products for special event planners (Uber Events and Lyft Events) where planners can purchase ride passes and distribute to guests. Despite the popularity, the benefit of ridesharing services from a planner's perspective is still under discussion. Empirical studies (Alexander and González 2015; Li, Hong, and Zhang 2016; Rayle et al. 2016) show that ridesharing services drastically reduce traffic congestion in urban areas and expands mobility options for cities. However, in a PSE network, these conclusions need to be carefully evaluated. On the one hand, ridesharing option would reduce the parking demand during a special event. On the other hand, extended mobility might attract more attendees to a special event. And the intense lane-changing and waiting behavior at drop-off locations could worsen the traffic condition.

Lane-reversal is an effective strategy to accommodate unbalanced travel demand during a special event. The effectiveness of implementing lane reversal for unbalanced demand has been extensively studied in existing literature (Kim, Shekhar, and Min 2008; Xie, Lin, and Travis Waller 2010). In a more general context, Wolshon and Lambert (Wolshon and Lambert 2004) reviewed lane-reversal techniques as well as its engineering practices. During the ingress of a PSE, reversing the direction of outbound lanes could help enhance inbound capacity.

To summarize, in this dissertation we investigate congestion mitigation during a special event from both travelers and planners' perspectives. First, a parking occupancy

prediction framework is presented in Chapter 3. It is the key input to many parking related models and the fundamental information for travelers or planners to make parking-related decisions during a special event. The prediction power of the framework is tested by real data from recurrent and non-current special events. Chapter 4 investigates the idea of introducing learning algorithms into parking guidance and information systems that employ a central server, in order to provide estimated optimal parking searching strategies to travelers. This approach could minimize the expected travel time for individual travelers using the service, but it might not be beneficial to the whole network. From the planners' perspective, a systematic way to generate comprehensive PSE traffic plans is proposed in Chapter 5 with the consideration of parking, ridesharing and route recommendations at the same time. It is followed by the conclusion and future search in Chapter 6.

3 A PARCTICAL MODEL-BASED FRAMEWORK FOR PREDICTING PARKING AVAILABILITY

3.1 Introduction

The parking availability information is highly valued by travelers, and is one of the most important inputs to many parking models. The research communities have been studying various aspects of parking for decades. As early as in the 1950s, Vickrey has suggested time-varying parking fees to ensure the economical utilization of parking spaces (Vickrey 1994). To this day, policy and operational innovations aiming to improve social welfare is still a predominant topic in parking research. Such innovations include pricing (Arnott et al., 1991; Zhang and van Wee, 2011; Qian and Rajagopal, 2014; Mackowski et al, 2015; He et al, 2015), reservation (Teodorovic and Lucic, 2006; Boehle et al., 2008; Delot et al., 2009; Caicedo et al., 2012; Liu, et al., 2014; Shin and Jun, 2014; Chen et al., 2015), and information provision (Waterson, et al., 2001; Thompson et al., 2001; Caicedo, 2009; Sun et al., 2016). Driven by the modeling need in these work, many studies have specifically investigated how parking competition affects both temporal and spatial travel patterns. The former is commonly achieved through bottleneck models (Yang et al, 2013; Xiao et al., 2016) while the latter through non-atomic games (more commonly referred to as traffic assignment) (Bifulco, 1993; Arnott and Rowse, 1999; Anderson and de Palma, 2004; Boyles et al., 2015) or more general game-theoretical framework (Ayala et al, 2011; Kokolaki et al., 2012; Guo et al., 2013). In addition, agent-based simulation employing

rule-based or discrete choice models (Beneson, et al, 2008; Waraich and Axhausen, 2012) has also been commonly adopted. It is worth noting that although information provision, more specifically the provision of parking occupancy and probability of finding a parking spot, is not as extensively studied as other parking management strategies, it has attracted more attention recently and are being considered by some of the models as key components, inputs, or assumptions (Caicedo, et al., 2012; Shin and Jun, 2014; Qian and Rajagopal, 2013; Guo et al., 2013; Schlote et al., 2014; Boyles et al., 2015; Chen et al., 2015). Additionally, it has been found in a recent empirical study (Chaniotakis and Pel, 2015) that the availability of parking after a reasonable search time does significantly affect drivers' parking choices.

In consideration of the significance of future parking availability information in parking searching, we propose a model-based practical framework to predict future occupancy from historical occupancy data alone. The framework consists of two modules: estimation of model parameters, and occupancy prediction. At the core of the predictive framework, a queuing model is employed to describe the stochastic occupancy change of a parking facility. While the underlying queuing model can be any reasonable model, we demonstrate the framework in this dissertation with the well-established continuous-time Markov M\M\C\C queue. The possibility of adopting a different queuing model that can potentially incorporate the parking-searching process is also discussed. The parameter estimation module and the occupancy prediction module are both built on the underlying queuing model. To apply the estimation and prediction methods in the real world, a few

practical considerations are accounted for in the framework with methods to handle variations of arrival and departure patterns from day to day and within a day, including special events. The proposed framework and models are validated by both simulated and real data. Our San Francisco case studies demonstrate that the parameters estimated offline can lead to accurate predictions of parking facility occupancy both with and without real-time updates. We also performed extensive numerical experiments to compare the proposed framework and methods with several pure machine-learning methods in recent literature. It is proved that our approach delivers equal or better performance, but requires a computation time that is orders of magnitude less to tune and train the model. Additionally, our approach can predict for any time in the future with one training process, while pure machine-learning methods have to train a specific model for a different prediction interval to achieve the same level of accuracy. The framework and the methods are validated using both simulated and real-world data. Our San Francisco case studies demonstrate that the parameters estimated offline can lead to accurate predictions of parking facility occupancy both with and without real-time occupancy. Even when the underlying model may be inaccurate due to limited data availability (such as parking occupancy data during special event), a potential drawback of model-based approaches, the prediction module can be easily modified to offer prediction that is more accurate. We also performed extensive numerical experiments to compare the proposed framework and methods with several pure machine-learning methods published in recent literature. It is found that our approach delivers equal or

better performance, but requires a computation time that is orders of magnitude less to tune and train the model. Additionally, our model can predict for any time in the future with one training process while pure machine learning methods have to train a separate model for each time window of interest to achieve accurate prediction. Moreover, by adopting a model-based approach over pure statistical and machine-learning methods, the framework could potentially incorporate the parking searching process (which can be reflected in the stochastic arrival process in the queuing model) in parking occupancy prediction. This would enable investigation on network traffic effects as a result of parking availability information provision and the subsequent possibly altered parking search behaviors using game-theoretical approaches.

To this end, the contributions of this chapter include: 1) a model-based practical parking occupancy prediction framework grounded in the underlying stochastic queuing process; 2) the first to validate a model-based parking occupancy prediction framework using real data; and 3) insights on the applicability and suitability of the proposed model-based framework from comparison with pure machine learning parking occupancy prediction methods.

3.2 Related Work

Two approaches to parking availability prediction exist in the literature.

One approach starts with establishing some underlying model for the parking process; and parking availability prediction relies on the estimation of model parameters. The

underlying model usually explicitly considers the stochastic arrival and departure processes of a parking facility. The arrival process is commonly assumed Poisson. As for the departure process, it is often assumed the parking duration follows negative exponential distribution (Millard-Ball et al., 2014; Portilla et al., 2009) or discretized Gamma distribution (Caicedo et al., 2012). Caliskan et al (2007) adopt the classic continuous-time birth-and-death queueing model with Poisson arrival and exponential inter-departure time. In their simulation work, they assumed that the departure rate is given, and employed a maximum likelihood approach to estimate the arrival rate from simulated occupancy data. They further predicted future occupancy with the estimated arrival rate and the given departure rate with an autoregressive Gaussian process. Similarly, Lu et al. (2009) estimate the probability that a parking facility is completely full using the same model but assumes that advanced technologies are in place to provide data on arrival and departure rates directly. Boyles et al. (2015) incorporate the same continuous-time Markov model to specify the probabilistic parking availability in their network equilibrium analysis. Caicedo et al. (2012) proposed a centralized system for parking request allocation that has parking availability forecast as one of its components. The system computes future parking availability forecast based on requests allocation and simulated parking durations from discretized Gamma distribution. Existing studies in this category are mainly theoretical, and have at best validated their models using simulation. In addition, most of the aforementioned studies are not able to estimate parameters for the

arrival and departure processes simultaneously, and require additional assumptions for prediction.

Another approach, instead of modeling the underlying parking process, applies statistical and machine learning methods to predict future occupancy directly from the observed data. These methods include simple regression (McGuiness and McNeil, 1991; Burns and Faurot, 1992), database system (Dunning 2006), chaotic time series analysis (Liu et al, 2010), multivariate spatiotemporal regression (Rajabioun and Ioannou, 2015), neural network (Vlahogianni et al, 2015; Ji et al, 2015; Ziat et al, 2016; Tiedemann et al, 2015), and clustering (Tiedemann et al., 2015; Tamrazian et al., 2015). One drawback of these models is the extensive tuning of the model structure. For example, in neural network approaches, model structure parameters include the number of input nodes, the number of input layers / hidden nodes, learning rate and momentum. Extensive tuning is required to determine these parameters before the actual learning of weights of each input and hidden nodes can take place. Moreover, while these methods are able to provide accurate prediction for a given time window and can be very useful in implementations of parking information systems, their value in exploring innovative parking policies and operational strategies is limited. This is because such statistical and machine learning algorithms are isolated from other aspects of the transportation system; and they are not set up to account for changes in human behaviors and traffic flow in the entire network, which are ultimately reflected in the stochastic arrival and departure processes.

The predictive framework and methods developed in our work fall into the first approach, where the underlying stochastic parking process is explicitly modeled. Different from previous work in this category, we have validated our approaches using both simulated and real data. Moreover, our approach is also able to estimate at the same time both arrival and departure rates based on historical occupancy data alone. While our predictive framework also employs regression techniques under certain conditions, the governing equation (model structure) is derived from the analytical properties of the underlying Markov queueing model and does not require tuning.

3.3 Prediction Framework

The prediction framework consists of a parameter estimation module and an occupancy prediction module. At the core of the predictive framework, a queuing model is employed to describe the stochastic occupancy change of a parking facility. While the underlying queuing model can be any reasonable model, we demonstrate the framework in this dissertation with the well-established continuous-time Markov M\M\C\C queue. Mathematical properties of the M\M\C\C model, such as the time-dependent expectation, together with its asymptote, and the time-dependent variance of the parking facility occupancy are derived analytically under specific conditions. Applying the underlying queuing model, two approaches are developed to estimate the arrival and departure rates from historical occupancy data. The first approach makes use of the analytical properties of the occupancy probability distribution and employs curve-fitting techniques; and is

suitable when the parking facility is under-saturated. When the parking facility is over-saturated, the second approach applies maximum likelihood or least squares estimation directly. To apply any of the two base estimation methods in the real world, a few practical considerations are further discussed and methods proposed to handle arrival and departure patterns that are more complex. For example, due to day-to-day variations in the arrival and departure patterns, data clustering should be applied first to arrange historical daily occupancy time series into groups. When analyzing time-series data of occupancy within a 24-hour timeframe, the time points that may mark a change in the underlying arrival and departure rates need to be identified; and an iterative process is proposed in this dissertation for this purpose. A simple prediction method based on the analytical properties of the occupancy probability distribution can be applied for normal parking traffic. On the other hand, a modified prediction method is proposed to offer more accuracy during special events, for which data availability might be limited and the underlying parking searching process (and thus the arrival and departure patterns) is not well understood. The prediction module can be engaged both with and without real-time update, and does not require additional data other than observed occupancy at the beginning of the day or in real time.

3.3.1 Stochastic Queueing Model for Parking Process

In this section, we first provide a brief introduction of a widely used (Caliskan, 2007; Lu et al., 2009; Boyles, 2015) continuous-time Markov model (M/M/C/C queueing model)

for the stochastic parking process of a single parking facility. Time-dependent analytical properties of the model are then derived.

It is worth mentioning that the underlying queuing model can be any reasonable model. One possible alternative is to consider a discrete-time model where both the arrival and departure are described by a binomial process. This could potentially allow us to incorporate the parking-searching process by treating the probability that an approaching vehicle would select a given parking facility as a function of traffic and parking facility characteristics (and possibly driver characteristics). The discrete model could also potentially be more computationally efficient. Please see Appendix B for more discussions.

3.3.1.1 M/M/C/C Queueing Model

The state of the queue corresponds to the occupancy of a parking facility, defined as the number of vehicles currently parking in it. Note that parking occupancy is a discrete random variable. Denote the time-dependent probability distribution of the parking occupancy as $O(t)$, a row vector whose elements represent the probabilities of each occupancy value, from 0 to the capacity C of the parking facility. $O(t)$ is the product of initial occupancy $O(0)$ and the Markov transition-probability matrix $P(t)$.

Both arrival and departure are assumed to follow the Poisson process. The arrival rate of the parking facility is denoted as λ, and the parking time of each parked vehicle follows a negative exponential distribution with a mean value of $1/\mu$ units of time. If the current occupancy is n, then the current departure rate of the entire parking lot is $n\mu$. As

described above, the parking process is in fact a birth-and-death process (Ross, 2014), more specifically a Poisson queuing process with finite servers (equal to the capacity of the parking facility C) and zero buffer size. For fixed λ and μ, it is well established that the Kolmogorov's backward equations can be used to solve for the transition probability matrix $P(t)$ (Ross, 2014).

3.3.1.2 Model Properties

This section further investigates the time-dependent model properties, such as expectation, variance, and asymptote, before applying them for parking occupancy prediction.

Let N_t denote the occupancy at time t. Assume that during a period $[t, t+h]$ where $N_t < C$ always holds; i.e., the parking facility is under-saturated. Then:

$$E(N_{t+h}|N_t) = N_t + (\lambda - N_t\mu)h$$

We have proved that (see Appendix A):

$$E_t = e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu} \tag{3.1}$$

It is evident from Equation (3.1) that when the facility is under-saturated, the occupancy expectation will converge to λ/μ after a sufficiently long time. Furthermore, equation (3.1) will degenerate to λt when the departure rate is 0. In addition to the expected occupancy, the variance of N_t can also be written as a simple function of λ, μ, t, if the parking facility is under-saturated (See Appendix A for detailed derivation):

$$V_t = e^{-2\mu t}(V_0 - E_0) + e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu} \tag{3.2}$$

Equations (3.1) and (3.2) provide the basis for our parking occupancy prediction framework when the parking facility is under-saturated. First, curve-fitting techniques can be employed offline to estimate the parameters from time-series historical occupancy data. When multiple time series are available (for example, time-series occupancy data during peak hour from multiple days), the historical time-dependent mean and variance of the occupancy can be calculated to serve as the observation. Any of the expectation or variance equations would allow us to estimate both the arrival and departure rate at the same time without additional assumptions or data requirements. Next, if current occupancy is available, expected future occupancy can be predicted from equation (3.1) with the estimated arrival and departure rate. It should be pointed out that when the facility is over-saturated, Equations (3.1) and (3.2) no longer hold. In this case, parameter estimation is much more difficult. We propose to adopt maximum likelihood or least squares estimation in conjecture with the Markov model directly to estimate the parameters. The two estimation approaches are discussed in the next section.

3.3.2 Estimating Model Parameters from Time-series Data

In this section, two methods are proposed for two different situations to estimate the parameters in the Markov models described in Section 3.3.1.1.

Regression Parameter Estimation:

This method is applicable to the case where the capacity of a parking facility is not exceeded, i.e., the facility is under-saturated. Parameters for both the arrival and departure processes can be efficiently estimated at the same time with historical / time-

series data of occupancy alone. Compared to pure statistical or machine learning methods, the governing regression equation in our approach is derived from analytical properties of the Markov models, and does not require additional tuning of the model structure (e.g., number of input nodes, number of hidden nodes, etc. in neural network approaches).

Direct Maximum Likelihood or Least Squares Parameter Estimation: This method can be used more generally, even when the parking facility is over-saturated. However, it is more computationally demanding. When available historical data is limited, it is recommended to fix one of the arrival and departure rates in order to obtain an accurate estimate. This is similar to the requirement in most previous studies (Caliskan et al., 2007; Caicedo et al., 2012; Portilla et al., 2009).

Numerical experiments with simulated data are first performed to demonstrate the two methods above. Case studies with real data are presented in Section 3.5.

3.3.2.1 Parameter Estimation Methods

Regression with Derived Expectation or Variation Equation

Suppose a parking facility is under-saturated, i.e., no arriving vehicle is rejected because the facility is full, and the arrival and departure parameters are constant during a time window $[t_s, t_e]$, where t_s is the starting time and t_e the ending time. Note that in practice, the observed occupancy is commonly in time series form. Suppose $t_e = t_s + m\Delta t$ and let $\bar{n}_{t_s+i\Delta t}$ denote the mean of the observed occupancy at time point $t = t_s + i\Delta t, \forall i = 0,1,2, \dots m$ over multiple days. Applying Equation (3.1), a constrained regression problem can be formulated to estimate (λ, μ).The objective is to minimize the square of the

difference between predicted and observed mean occupancy values $Min_{\lambda,p}\ \Sigma_{i=1}^{m}\left(y(i) - \bar{n}_{t_s+i\Delta t}\right)^2$, where $y(i) = e^{-\mu i\Delta t}\left(\bar{n}_{t_s} - \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu}, \forall\, i = 1,2,\dots m$. This forces the fitted curve to always pass through the first data point $\left(t_s, \bar{n}_{t_s}\right)$ in the time window, and eliminates error propagation when multiple time windows are being considered. The constraints include 1) the average parking duration should be greater than one time interval, an essential requirement of the parking model; 2) the values of λ and p are realistic, within the physically possible ranges of entrance and exit rates.

Maximum Likelihood or Least Squares Estimation with Markov Model

When the parking facility is over-saturated, equations (3.1) and (3.2) no longer hold and the regression method is no longer applicable. In this case, we propose to adopt appropriate optimization techniques directly to seek a pair of (λ, μ) for the proposed Markov model that maximizes the likelihood of observed occupancy values or minimizes the sum of error squared between observed and predicted occupancy distributions.

Suppose historical occupancy data at t_1, t_2 $(t_1 < t_2)$ from M days are available. Denoted the observed occupancy values at t_1 over M days as $n_1^1, n_1^2, \dots n_1^k, .., n_1^M$ and those at t_2 as $n_2^1, n_2^2, \dots n_2^k, .., n_2^M$. From these observations, the historical occupancy distributions can be calculated as $O_1 = \left(f_1(0), f_1(1), \dots f_1(C)\right)/\sum_{k=1}^{M} n_1^k$ and $O_2 = \left(f_2(0), f_2(1), \dots f_2(C)\right)/\sum_{k=1}^{M} n_2^k$, where $f_t(i)$ is the frequency of occupancy value i being observed at time t. Using O_1 as the starting point, and employing the proposed Markov model $O(t) = O(0) \cdot P(t)$ with any given pair of (λ, μ), we can calculate the

predicted occupancy distribution at t_2, denoted as $\tilde{O}_2$. The maximum likelihood estimation seeks a pair of (λ, μ) that maximizes the term $\prod_{i=0}^{C} \tilde{O}_2(i)^{f_2(i)}$, the likelihood of the actual observations at time t_2 under the predicted occupancy distribution $\tilde{O}_2$. The least squares estimation, on the other hand, seeks a pair of (λ, μ) that minimizes the term $\sum_{i=0}^{C}\left(O_2(i) - \tilde{O}_2(i)\right)^2$.

It is worth noting the time-dependent occupancy distribution from the continuous-time model is difficult to evaluate in general. It involves evaluation of a matrix exponential operator $e^X := \sum_{k=0}^{\infty} X^k/k!$ (Coddington, 1955), which is non-trivial and does not have a closed-form expression when the arrival or departure rate varies with time. Other reasonable queuing models can also be adopted in the framework. Appendix B discusses one such alternative, a discrete-time model, which could potentially be more computationally efficient.

3.3.2.2 Numerical Examples

To demonstrate the occupancy prediction framework, simulation experiments are performed. Simulated arrivals and departures are generated based on the parking model, and the resulting occupancy values are treated as the observed historical data. Note that since the process used to generate the random data is identical to the one used in the estimation approaches, the numerical examples in this subsection do not necessarily validate the proposed models for real-world application. Rather, the simulated numerical examples explore the applicability of the two parameter estimation methods in under- and

over-saturated conditions. The appropriateness of the proposed methods in real-world applications is demonstrated in Section 5.

In the simulation, the capacity of the parking facility C is set to 20. The simulation duration is 50 time intervals. The average arrival rate λ is 1 vehicle per time interval, and the departure rate μ during any given interval is set to 0.05 (equivalent to an average parking time of 20 time intervals). The starting occupancy distribution has non-zero probabilities for occupancy values 2 – 6 as 0.1, 0.3, 0.3, 0.2, and 0.1, respectively. From simulated arrivals and departures over 100 simulation runs, the distribution of occupancy (Figure 3-1) and the expectation of occupancy (Figure 3-2) over time are computed to serve as historical data.

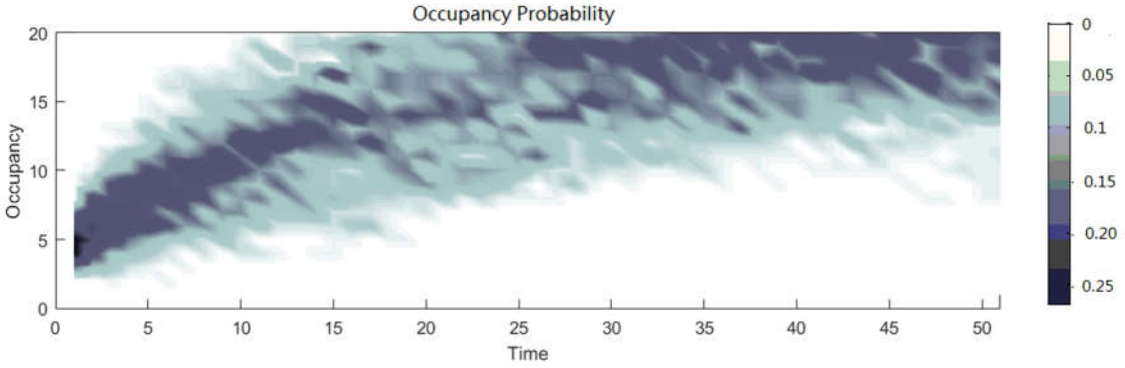

Figure 3-1 Occupancy Distribution Over Time from Simulated Data

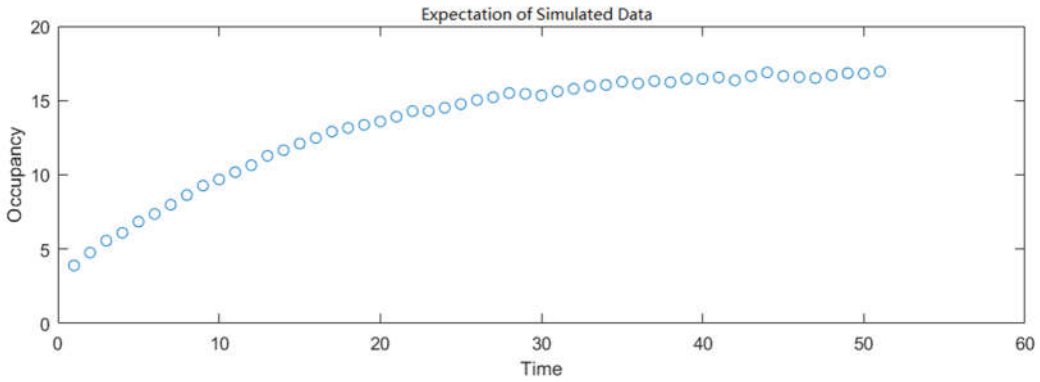

Figure 3-2 Occupancy Expectation Over Time from Simulated Data

Under-saturated Facility

From Figure 3-1, it can be seen that the occupancy rarely reaches capacity when $t \leq 20$. Therefore, we can apply the regression method for the period $[1, 20]$. Figure 3-3 shows the fitted expectation curve. The fitted parameters $(\lambda, \mu) = (1.0175, 0.0521)$ from 20 data points are very close to the simulation input. It is worth mentioning that gate constraints are not explicitly enforced in this simulated experiment.

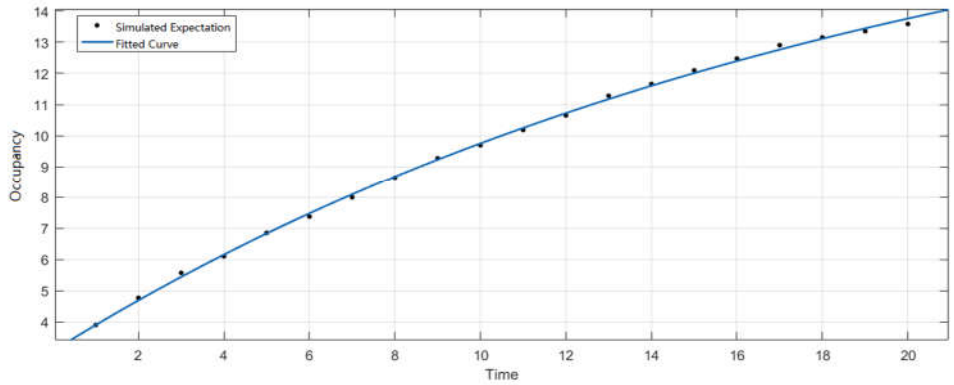

Figure 3-3 Curve Fitting for t∈[1,20]

In a real application, arrival and departure parameters may not remain constant for an extended period, which means that there might only be a limited number of data points for curve fitting. To see how the regression method performs when there are fewer data points, an expectation curve is fitted with the first 5 data points only (Figure 3-4). The estimation result is $(\lambda, \mu) = (1.0322, 0.0533)$, still quite close to the simulation inputs. This indicates that regression method could work well even with fewer data points.

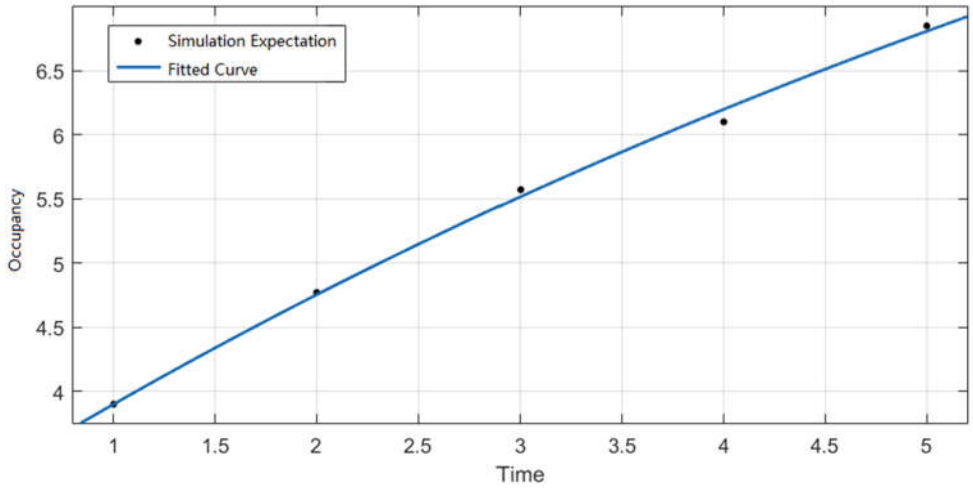

Figure 3-4 Curve Fitting for t∈[1,5]

We also performed an estimation using the variation curve Equation (3.2) and obtained very similar results.

Over-saturated Facility

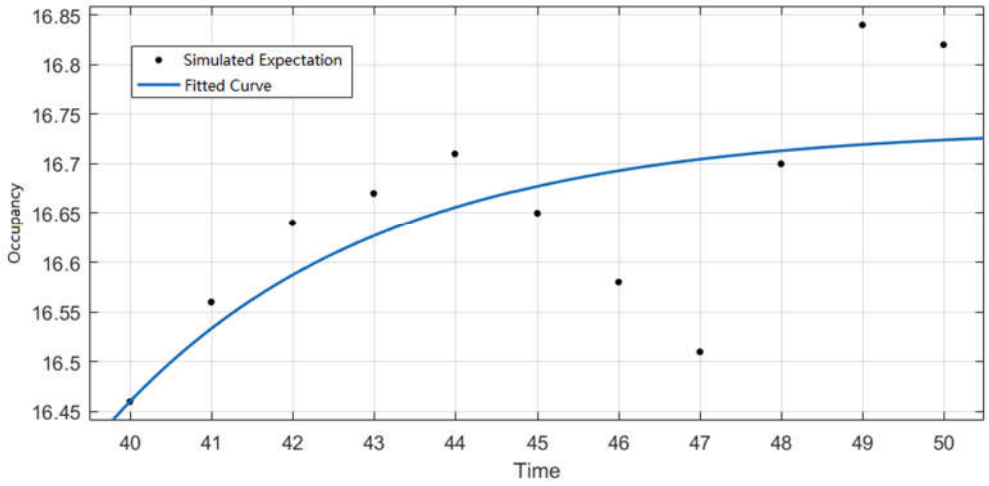

Figure 3-5 Curving Fitting for t∈[40,50]

In this simulation experiment, there is a high probability that the parking facility is fully occupied as t increases (Figure 3-1). In this case, the expectation curve Equation (3.1) no longer holds. If the regression method is applied nonetheless, the results would be inaccurate. The estimated $(\lambda, \mu) = (2.7833, 0.1645)$ using data from $t \in [40, 50]$ (Figure 3-5) is quite different from the simulation input.

Table 3-1 Simulation-based Optimization Estimation Results for $t \in [40,50]$

Fixed departure rate $\mu = 0.05$ Estimate λ			Fix arrival rate $\lambda = 1$ Estimate μ		
Starting Point	Maximum Likelihood	Least Square	Starting Point	Maximum Likelihood	Least Square
0	0.9974	0.9973	0	0.0486	0.0494

0.1	0.9973	0.9973	0.2	0.0493	0.0500
0.2	0.9973	0.9973	0.1	0.0488	0.0488
0.5	0.9974	0.9973	0.067	0.0500	0.0500
1	0.9974	0.9973	0.05	0.0500	0.0500

Instead, direct maximum likelihood and least squares estimation is implemented with a compass search algorithm (Kolda et al., 2003) to seek the best feasible pair of (λ, μ). Both λ and μ can be estimated simultaneously rather accurately, if the observation data set is large (for example, with 10000 time series). However, when the size of historical data is limited (such as in our case with 100 time series), the direct optimization may converge to a local optimal depending on the starting point of the compass search. In this case, it is ideal if one parameter can be fixed based on prior knowledge. For example, with gated parking facilities, the arrival rate can be obtained from gate readings. Table 3-1 Simulation-based Optimization Estimation Results for $t \in [40,50]$ reports the estimation results from simulation-based optimization with various starting values and different objectives, when one of the two parameters is fixed. It can be seen the estimated values are quite accurate, regardless of the starting point.

3.3.3 Parameter Estimation in Real-world Applications

Before applying the regression method, we need to identify the days from all available data when drivers have similar activity patterns that will result in a similar arrival rate and parking time. With those groups defined, a parking facility is considered to

experience the same arrival and departure rate at the same time of day for any day in a given group. On the other hand, it is important to determine the time of day when λ and μ change. Clustering techniques can be employed for the former; and an iterative approach can be adopted for the latter.

3.3.3.1 Day-to-day Patterns

To avoid over complicating and dividing historical data into too many groups, we propose to only consider two to three groups of days, using between-group linkage (Murtagh, 1983) hierarchical cluster method in SPSS (IBM, 2016). Possible classifications include "workdays" versus "holidays", or "regular workdays" versus "high demand days" versus "holidays".

Between-group linkage hierarchical cluster is a common clustering method. Each sample is initially treated as a separate group. The gaps between every pair of groups are calculated and the two closest groups are combined into one. The gaps between groups are defined as the average distance between each pair of data points in different groups; and the distance between data points is defined as the Euclid distance. This process is repeated until the desired number of groups is achieved.

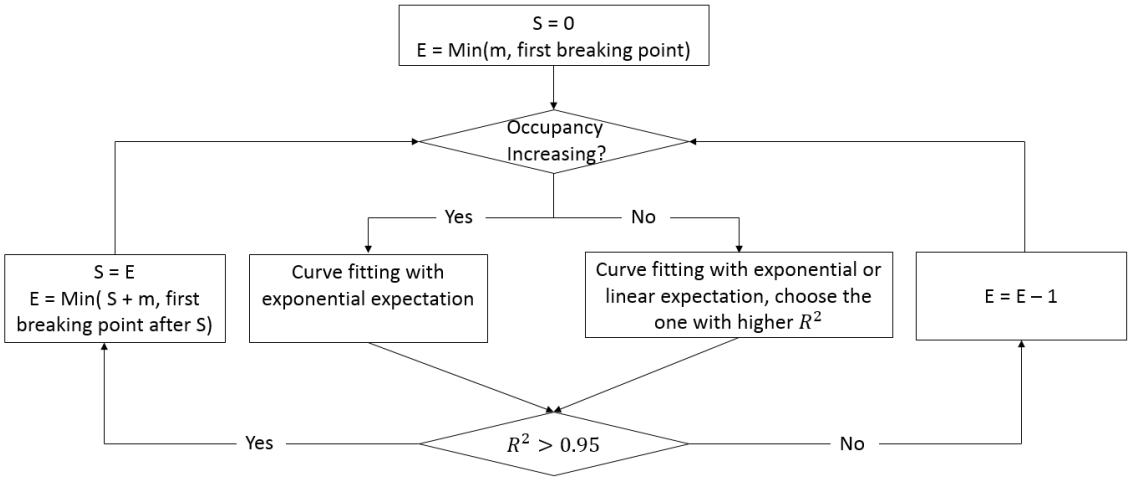

Figure 3-6 Iterative Estimation Framework

3.3.3.2 Within-day Patterns

We define the time of day when λ and p change as breaking points. Apparently, (λ, μ) must change at a local maximum or local minimum of the historical mean occupancy curve. These breaking points divide the total analysis period into multiple periods within which the historical expectation curve is monotone. For a period where the historical mean occupancy decreases, it is obvious that $\mu > 0$; and the least square estimation from 3.2.1 should be applied for parameter estimation. When historical mean occupancy monotonically increases (μ may or may not be zero), both the original exponential and degenerated linear formulations can be used to fit the data and the one with better performance chosen. Furthermore, since parameters normally do not remain constant for an extended period in real world, we let m represent the largest number of time intervals that (λ, μ) could remain constant, and start our regression with a time window of at most m data points. If the resulting R^2 is sufficiently large (greater than a pre-defined value),

move to the next period. Otherwise, reduce the length of the period by one time interval at a time until R^2 meets the criteria. A flowchart of the process is shown in Figure 3-6 Iterative Estimation Framework

3.3.4 Occupancy Prediction in Real-world Applications

Once the time-dependent arrival and departure rate and are estimated, they can be applied to predict future occupancy using Equation (3.1). The prediction module can be engaged both with and without real-time updates, and does not require additional data other than observed occupancy at the beginning of the day or in real time. Additionally, it would be beneficial to monitor the prediction error in real-time, although not required, in order to make any adjustments to capture abnormal parking demand caused by special events or other unforeseen reasons that may not be reflected otherwise in the historical data set.

3.3.4.1 Prediction with and without Real-time Updates

The model-based prediction method proposed allows occupancy prediction for any time in the future both with and without real-time data.

Without any real-time occupancy data, prediction of the next time interval can be performed repeatedly based on the output from previous occupancy prediction. For example, a 24-hour prediction without real-time data predicts the occupancy for the entire day based on the observed occupancy at the beginning of the day. It is certainly limited in handling abnormal arrival and departure patterns, but has its use in planning and other transportation analysis.

We can also update the prediction based on real-time data as they become available. The prediction interval therefore could be significantly shorter, bounded only by the data collection interval. Note that this does not mean that the only prediction available to users is from the previous time interval. Rather, predicted parking occupancy is available whenever a user queries the information, be it at the beginning of the trip or anytime during the trip. The real-time updates allow travelers to check for changes in occupancy prediction shortly before they arrive at their destination for a more accurate estimate.

3.3.4.2 Recurrent Over-saturation and Special Event

As described in Section 3.3.2.1, if a parking facility is recurrently over-saturated, a direct maximum likelihood or least squares estimation can be adopted. However, if a parking facility is only occasionally over saturated, possibly due to a special event, then the direct estimation methods would not be applicable. This is because the near- or over-saturation is not caused by recurring arrival and departure patterns, and there would not be sufficient historical data to support the direct estimation. There are multiple possible approaches to handle special event occupancy prediction, depending on data availability.

If rich historical special event data (days, times, durations, and types, etc.) is available, it is possible to apply clustering methods to divide the historical data into groups and learn the arrival rate and departure probability for each group. It is worth noting that care should be taken in accounting for the normal background parking demand during special events. For prediction, if information regarding the special event of question is known in advance, then one or more groups that have similar attributes could be selected to carry

out the feature-weighted average method (Tamrazian et al, 2015) and alike. Otherwise, multiple prediction threads based on different groups could be performed simultaneously, and their prediction errors monitored in real-time. The weight of each group can then be updated over time as a function of the prediction errors.

If the information on historical special event is insufficient, i.e., the underlying parking searching and queueing processes are not well-understood, reasonable predictions can still be achieved by refining the estimates based on real-time prediction error. When the actual occupancy becomes greater than the predicted value by a pre-defined threshold, it indicates that the historical arrival and departure rates are not able to capture the parking demand pattern of the moment, and that the ingress of a special event may be in process. The real-time prediction errors could serve as a simple piece-wise linear approximation to the additional special event arrivals. The occupancy prediction can be slightly modified by adding this component to the value predicted from the historical arrival and departure patterns. Note that in this approach, we do not need to know whether a special event is present or not, nor its time and duration.

3.4 Case Studies

Three case studies (two for regular workdays and one for a special event) with real occupancy data from San Francisco are presented to validate the performance of the proposed framework. These case studies were originally done with the discrete-time

model but as described above, the occupancy prediction will not be different for continuous-time model.

It is worth mentioning that for all the 14 parking facilities we collected data for, none was ever filled during our three-month data collection period. However, there were days where some parking facilities were very close to saturation due to a special event. In view of this, all three case studies in this section have adopted the regression method for parameter estimation. The prediction (both with and without real-time updates) for regular workdays is straightforward. The case studies show that the proposed model-based framework performs well both with and without real-time updates for a range of prediction time windows, where the paradigm shift from daytime to evening can be captured by the prediction with real-time updates. The special event case study demonstrates that the framework can achieve accurate prediction even if the underlying model may be inaccurate due to limited data availability.

3.4.1 Data Collection

A Hypertext Preprocessor (PHP) script was written to retrieve the San Francisco parking garage JavaScript Object Notation (JSON) file from Firebase's (Firebase, 2015) public data set from May 2015 to August 2015. A Cron job has been established to execute the PHP script every two minutes to keep bandwidth use low on the Firebase servers. The script parsed the garage name, the number of open spaces and the number of total spaces from the JSON file before writing the parsed data and a timestamp to a Comma Separated

Values (CSV) text file. The San Francisco garage parking data was retrieved with the JSON file. No street parking data was available during our data collection period.

3.4.2 Case Study 1

The purpose of this case study is to demonstrate all the components of the proposed framework. The case study is based on two weeks' worth of data from the Civic Center garage collected in May, 2015. Figure 3-7 plots the observed occupancy every 12 minutes for the two weeks. The missing points in the figures are due to the closure of the garage and lost connection to the database. These points are excluded from the occupancy expectation calculation. Due to the low occupancy between midnight and 6 am, we will focus on parameter estimation and occupancy prediction starting from 6 am. In view of the lack of any additional information on the maximum entering and leaving rates, we set both g_1 and g_2 to 60, i.e., it is assumed that at most 60 vehicles on average could enter or leave the parking garage every 12 minutes.

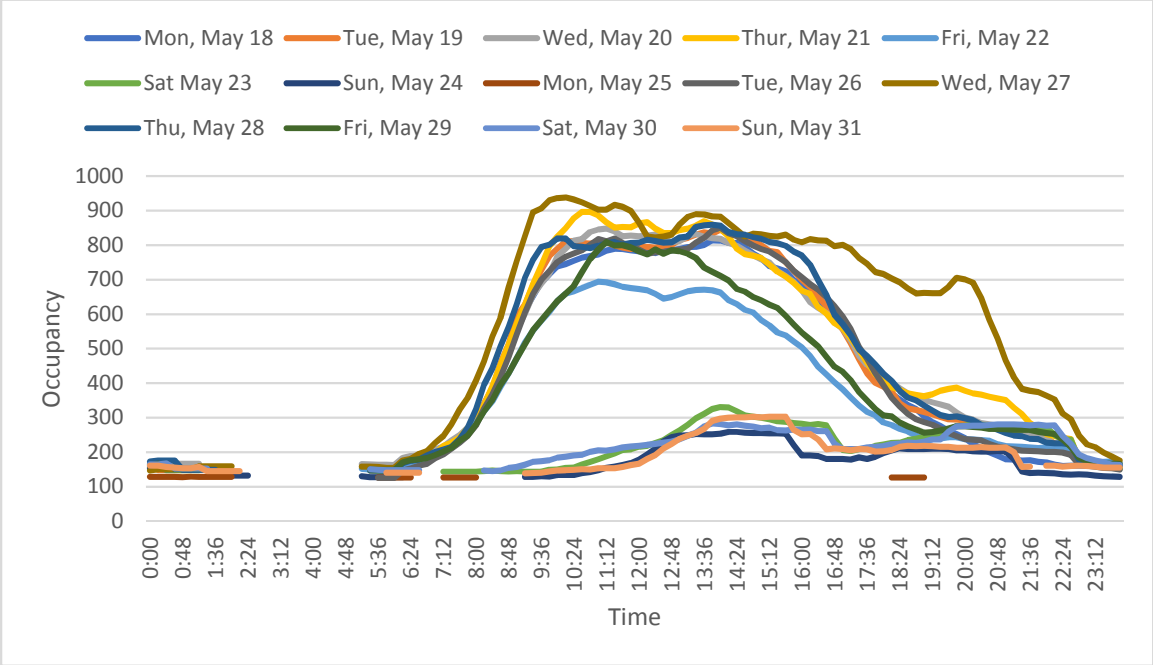

Figure 3-7 Occupancy Data (Civic Center Garage, San Francisco, May 18 – 31, 2015)

From hierarchical cluster analysis (Figure 3-8), the 14 days are first divided into two groups: "workday" and "holiday". Notice that May 25, 2015 is a weekday but is classified into the "holiday" group, because it was Memorial Day. Also, May 27 displays a significantly different occupancy pattern from other workdays, possibly caused by special events or randomness. Therefore, we excluded May 27, 2015 and used the rest of the days in the "workday" group as our analysis data set (a total of 8 days).

The expectation curve of the model (Equation (3.1)) is employed to estimate and predict "workday" group occupancy data for the Civic Center garage. The longest time window during which (λ, μ) remains constant is set to 1 hour in this case study. If R^2 of the

regression is smaller than 0.95, the time window will be reduced by 12 minutes until R^2 is sufficiently large.

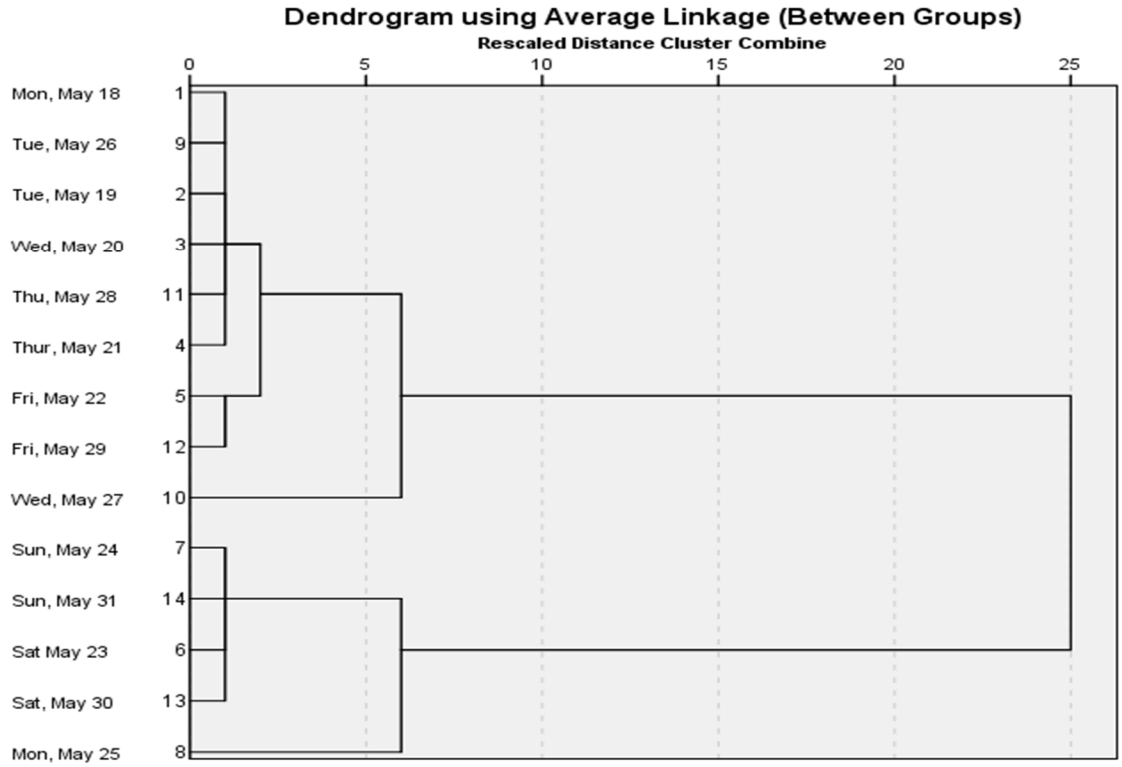

Figure 3-8 Hierarchical Cluster

Table 3-2 reports the estimated parameters between 13:00 and 15:36. From 13:00 to 14:00, the mean occupancy from the 8 days monotonically increases. Both models exponential (shown as Exponential in Table 3-2) and linear (shown as Linear in Table 3-2) formulation are employed; and it turns out that the Linear model, where the leaving probability is set to zero, fits the data better. Also note that the data did not support the assumption that the arrival rate is constant between 13:00 and 14:00. Rather, the final estimation results suggest that the arrival rate remained constant from 13:00 to 13:36, and

kept decreasing between 13:36 to 14:00. Starting from 14:00, the mean historical occupancy began to decrease, and the Exponential model should be applied. The results show that 15:00 is a breaking point where the leaving probability almost doubled.

With the estimated time-dependent arrival rate and leaving probability for workdays from the last two weeks of May 2015, we performed prediction every 12 minutes with and without real-time occupancy data (updated every 12 minutes). This case study uses testing data from June 2, 2015 to validate and demonstrate the performance of the proposed framework and methods.

Table 3-2 Part of Estimated Parameter Table

Interval	Starting Time	Historical Mean Occupancy	Exponential		Linear
			λ (# vehicles every 12 minutes)	μ (# vehicles every 12 minutes)	λ (# vehicles every 12 minutes)
1	13:00	811.5000	--	--	6.8
2	13:12	817.3333	--	--	6.8
3	13:24	828.8333	--	--	6.8
4	13:36	835.0000	--	--	3
5	13:48	838.0000	--	--	0.167

6	14: 00	838.1667	0	0.0133	Na
7	14: 12	823.1667	0	0.0133	Na
8	14: 24	814.0000	0	0.0133	Na
9	14: 36	802.8333	0	0.0133	Na
10	14: 48	792.0000	0	0.0133	Na
11	15: 00	781.8333	0	0.0233	Na
12	15: 12	767.5000	0	0.0233	Na
13	15: 24	755.0000	0	0.0233	Na
…	…	…	…	…	…
22	17:12	533.6667	54.2751	0.1883	Na
23	17:24	488.5000	54.2751	0.1883	Na
24	17:36	457.6667	54.2751	0.1883	Na
25	17:48	428.6667	54.2751	0.1883	Na
26	18:00	404.8333	54.2751	0.1883	Na
27	18:12	382.8333	80.9103	0.2723	Na
28	18:24	362.0000	80.9103	0.2723	Na
29	18:36	346.0000	70.82864	0.2384	Na
30	18:48	336.1667	70.82864	0.2384	Na
31	19:00	326.1667	70.82864	0.2384	Na
32	19:12	318.5000	0	0.0219	Na

33	19:24	311.8333	0	0.0219	Na
34	19:36	305.1667	0	0.0219	Na

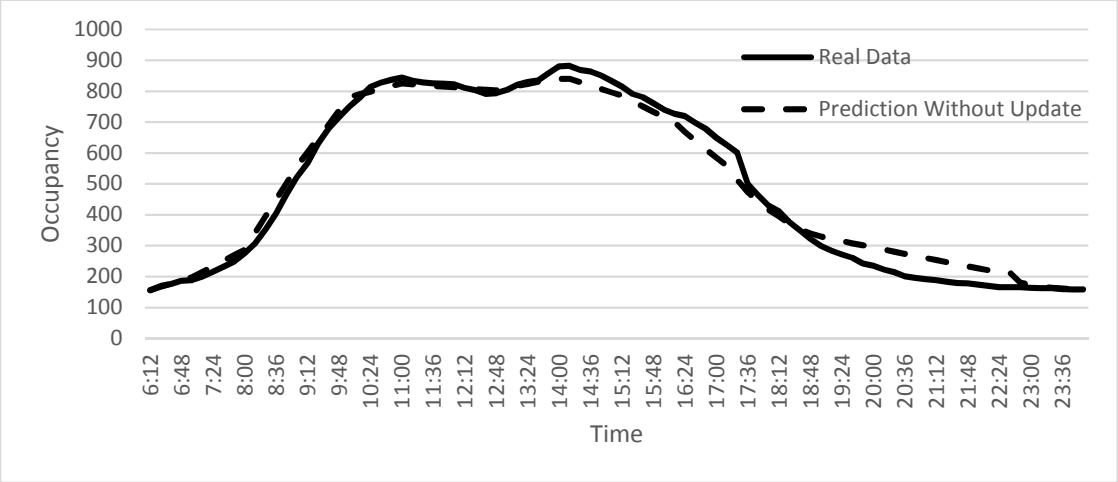

Figure 3-9 Prediction without Updates

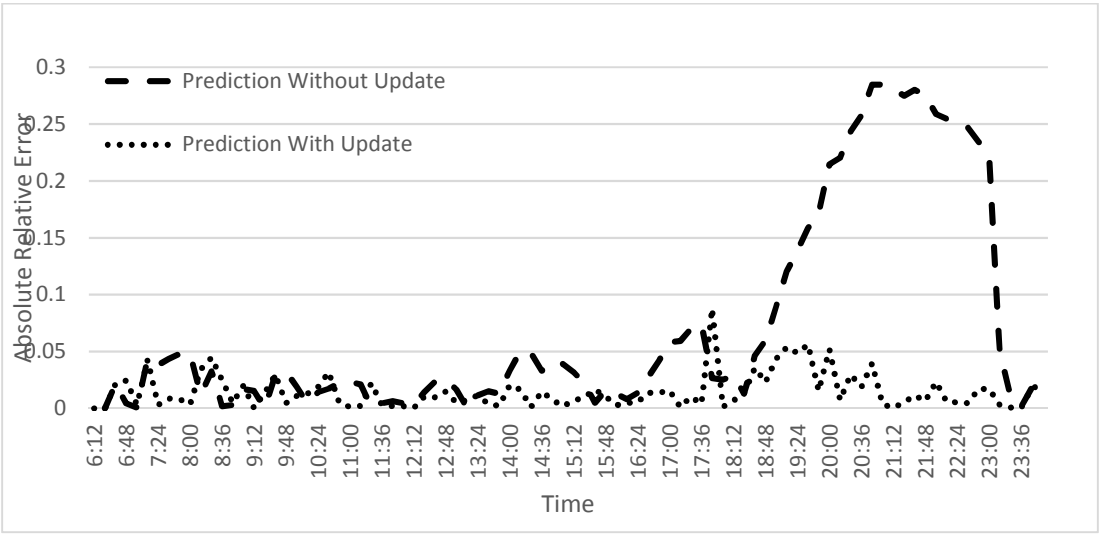

Figure 3-10 Absolute Relative Prediction Error with Updates

(Civic Center Garage, San Francisco, June 2, 2015)

The prediction without updates is based on the observed occupancy at 6:00 am. The results are presented in Figure 3-9 and Figure 3-10. The absolute relative error of the prediction is less than 10% from 6 am to 7 pm. From 7 pm to 10 pm, however, the occupancy is over-estimated (Figure 3-9), and the relative error is significant (Figure 3-10). A closer look at the event calendars of venues at Civic Center revealed that June 2nd, 2015 had no event after 5 pm while most of the days in the historical data set had at least one event after 7 pm[1]. This explains the over-estimation in the evening period for June 2nd, 2015, and also highlights the limitation of prediction without update.

With real-time data, an updated prediction is produced every 12 minutes in this case study. As can be seen from Figure 3-10, prediction with updates is able to capture the paradigm shift from daytime to evening, and produce estimates that are more accurate with a maximum absolute relative prediction error of less than 9% and a mean absolute relative error (MARE) of 1.486% for the entire day.

3.4.3 Case Study 2

The purpose of this case study is to examine the performance of the proposed model-based prediction framework when a parking facility is highly congested. Although none of the 14 parking garages was ever full during our data collection period, the data set does have days where a parking facility was close to saturation due to special events. June 19th, 2015 stands out for the Civic Center parking garage with abnormally high parking occupancy. It reached an occupancy of over 90% in the evening. The peak

[1] http://www.sfstation.com/calendar/san-francisco/civic-center/06-02-2015

occupancy is 781 out of a capacity of 843 at 10:12 pm. This indicates that the facility could be nearly- or even over-saturated on that day. Although the data shows that there were still some stalls available, they were not necessarily usable due to various reasons (reserved stalls, double-parking, undesirable surroundings such as low beams or support columns, etc.) Looking back, it is evident that the congestion was caused by the 100th celebration of San Francisco's City Hall on that day[2]. Since data on historical events of this scale is insufficient to perform the direct estimation method described in Section 3.3.2 or the clustering method described in Section 3.3.4.2, the modified prediction method as described in Section 3.3.4.2 is adopted for this case study.

We focus on a portion of the ingress window, which is from 6:48 pm to 10:12 pm, where the occupancy kept increasing. Since June 19th, 2015 is a Friday, data from the following three Fridays was used as our training dataset. We obtained the estimated arrival rate and departure probability for a "normal" Friday using the same estimation procedure as in the previous two case studies. The parameter estimation time interval Δt is 12 minutes, and the maximum time window during which (λ, μ) remains constant is also set to 12 minutes. Both the original and the modified prediction are performed, where the threshold is set to 10.

Figure 3-11 shows the predicted occupancy for the two prediction methods. Even without any training data from comparable special events, the linear approximation in the modified prediction method is able to handle the extra arrival well. Comparing to the

[2] https://www.sfstation.com/calendar/san-francisco/06-19-2015

original prediction, the modified method reduces the MARE during the study window from 5.23% to 1.74%, which is comparable to the MARE from the normal workday scenario in Case Study 1.

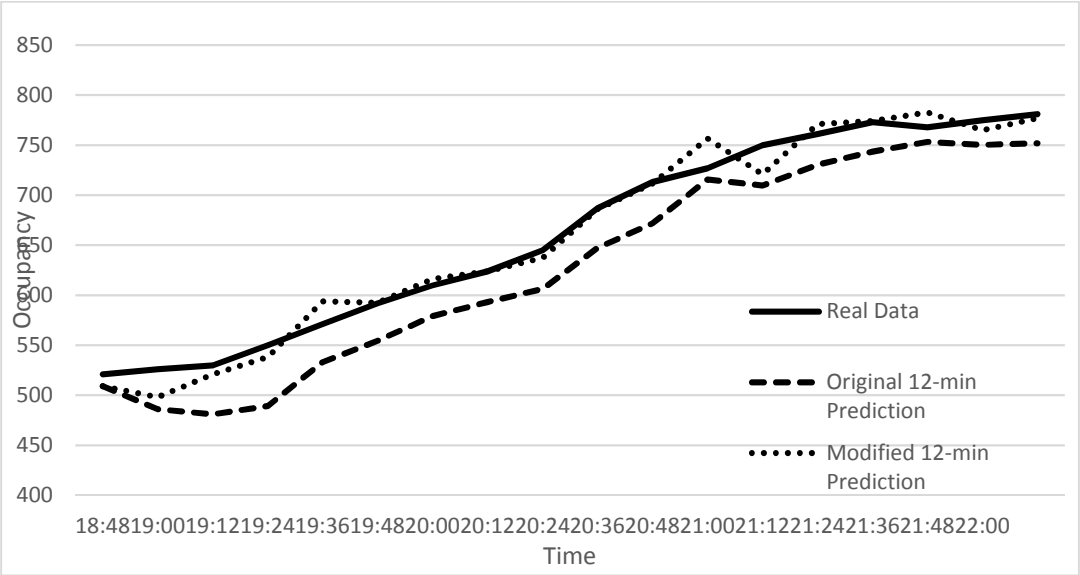

Figure 3-11 Modified vs. Original 12-min Prediction

(Civic Center Garage, San Francisco, Jun 19, 2015)

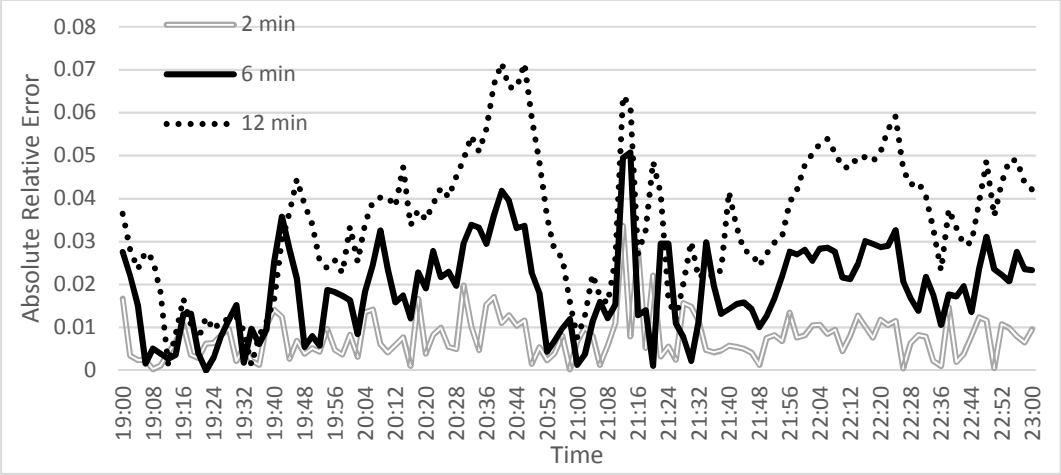

Figure 3-12 Absolute Relative Prediction Error for Three Different Prediction Intervals (Civic Center Garage, San Francisco, July 16, 2015)

3.4.4 Case Study 3

The purpose of this case study is to investigate how different prediction intervals affect prediction accuracy. The case study is based on historical data from July 6 to July 15, 2015. The procedure is the same as in case study 1. Since data was collected every 2 minutes during this period, the time interval Δt for parameter estimation is therefore set to 2 minutes. The maximum time window during which (λ, μ) remains constant is set to 40 minutes.

Occupancy prediction for 2, 4, 6, 8, 10 and 12 minutes in the future is performed based on the current real-time data. As expected, prediction error increases as the prediction interval gets longer. Figure 3-12 shows the errors of the predictions made 2, 6, and 12 minutes earlier. The evening period is selected because the errors are insignificant during the day even for the longest prediction interval. For prediction made 12 minutes ago, the

absolute relative error is no more than 8% during the evening period. The value reduces to 5% and 4% for predictions made 6 and 2 minutes ago, respectively. This suggests that while a short prediction interval leads to more accurate estimates, predictions made a while ago is still effective.

3.5 Comparison with Machine Learning Methods

To gain further insight into the applicability and suitability of the proposed model-based framework, this section compares the performance of our framework to those from three pure machine-learning methods found in recent literature. They include artificial neural network (ANN, Vlahogianni et al., 2015), wavelet neural network (WNN, Ji et al., 2015), and feature-weighted average method (Tamrazian et al., 2015). The former two are applicable to prediction with real-time update and the last is for prediction without real-time update only. These methods are selected primarily because they require the same data input as in our framework.

Vlahogianni et al. (2015) and Ji et al. (2015) both proposed to predict the occupancy for time interval $t + 1$ using real-time data from n past time intervals (including the current interval t). The general structure of their models is shown in Figure 3-13. The ANN (Vlahogianni et al., 2015) employs a sigmoid function at each hidden node to transform the weighted sum of all input nodes, whereas the WNN (Ji et al., 2015) uses Morlet function instead.

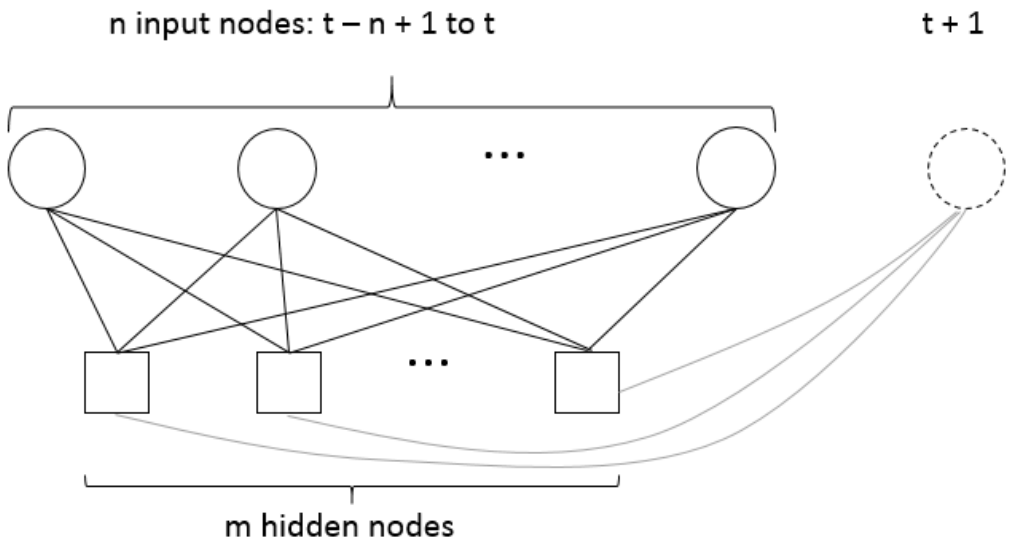

Figure 3-13 Neural Network Structure for Prediction of Interval t+1

The number of input nodes n, number of hidden nodes m, and the learning rate and momentum have to be determined before the actual learning of weights of each input and hidden nodes can take place. Vlahogianni et al. (2015) optimized the ANN structure using genetic algorithm with 5-fold cross-validation (Tan et. al, 2005). This process involves partitioning the original data randomly into five subsets of the same size. Given a neural network structure, it is evaluated five times, using each of the five subsets as testing data and the rest as training data for the weights of each input and hidden node. The average performance over the five evaluations of the neural network structure is used as the fitness measure in the genetic algorithm. Ji et al. (2015) did not provide details on how the parameters relevant to model structure are determined other than that they were selected by experiments.

The feature-weighted average method (Tamrazian et al., 2015) clusters the time series of daily occupancy into different groups. The prediction is performed by a weighted average of the means of each group, where the weights are proportional to the number of days with the same feature as the target day (day of the week, weather etc.) in different groups, and does not require real-time data.

Comparisons based on the case studies described in Section 3.4 show that our approach significantly outperforms the wavelet neural network and feature-weighted average methods in terms of prediction error. The artificial neural network is able to achieve the same level of accuracy comparing to our approach but requires hours to tune the model structure before training, whereas our approach only takes less than 30 seconds to estimate the arrival and departure parameters.

3.5.1 Prediction with Real-time Update for the Next Interval

This subsection first investigates the performances of ANN, WNN, and our model-based approach in predicting the occupancy for the next time interval, as proposed in Vlahogianni et al. (2015) and Ji et al. (2015). We trained separate ANN and WNN models for both case studies 1 and 2. For the ANN models, we implemented the same 5-fold cross validation genetic algorithm used in Vlahogianni et al (2015) to optimize the ANN structure, including the number of input nodes (number of previous intervals), the number of hidden nodes, learning rate and momentum. Following the same setting as in Vlahogianni et al (2015), 50 chromosomes and 100 generations are used in the genetic algorithm. Our implementation is based on Java with Weka API (Hall et al, 2009). On the

other hand, since it is not clear how exactly the WNN model was tuned in Ji et al. (2015) and no well-established package for WNN exists, we programmed a brute-force procedure in Matlab 2017a to determine the WNN structure. Once the neural network structure is determined, the weights for each input and hidden nodes are then trained using the full historical data sets used in Sections 3.4.2 and 3.4.3.

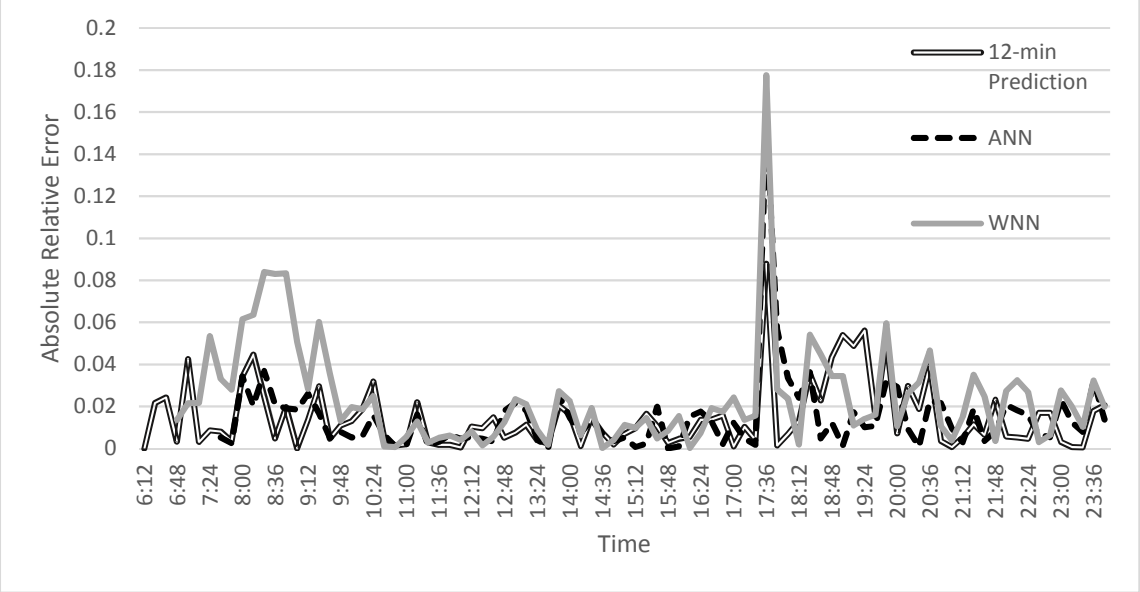

Figure 3-14 Relative Error of Three Methods for Case Study 1

Figure 3-14 shows the absolute relative error from ANN, WNN, and our approach for Case Study 1. It can be observed that our method performs considerably better than WNN (especially during the morning peak) and is comparable to ANN. It is worth noting that the errors from all three methods spiked when the daytime to evening transition occurred. For the overall performance, WNN leads to a MARE of 2.413%; ANN enjoys the lowest MARE of 1.464%; and our method is at 1.486% (see Table 3-3). It is not clear

whether the differences are statistically significant or not for this case study, since the number of data points in these time series is relatively small. However, while our approach took less than 30 seconds to estimate the arrival rates and departure probabilities, it took 8.5 hours to tune the structure of the ANN.

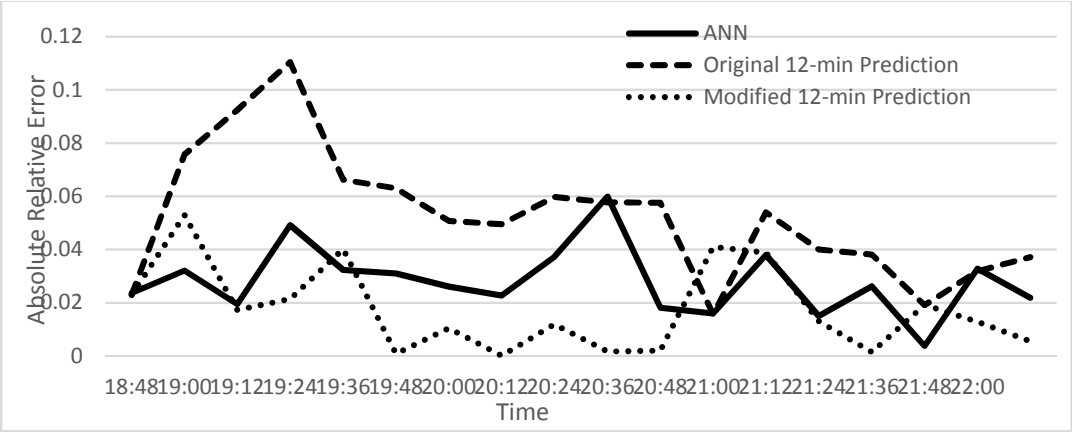

Figure 3-15 Relative Error of Three Methods for Case Study 2

Figure 3-15 plots the absolute relative prediction error from our approach (with both original and modified prediction) and ANN for Case Study 2. The same model tuning and training processes are employed. In this case study of a high profile special event, the training data is limited as data from comparable special events does not exist. Because of this, both our original prediction and ANN suffered from relatively large MARE compared to that in Case Study 1, where the time interval is also 12 minutes. However, with modified prediction, our approach is able to predict the occupancy more accurately with a MARE of 1.74%, much lower than ANN and the original prediction (See Table 3-3).

Table 3-3 MARE for Prediction with Update for the Next Interval

Approach		Case 1	Case 2
Proposed in This chapter	(w/ Original Prediction with update)	1.486%	5.23%
	(w/ Modified Prediction with update)	--	1.74%
ANN		1.464%	2.81%
WNN		2.413%	--

3.5.2 Prediction with Real-time Updated for Multiple Intervals in the Future

This section examines how our method compares with ANN when predicting for a future time point that is multiple intervals ahead. This is of practical importance because travelers are likely to query ahead of time in addition to obtaining updates right before arrival. In our approach, prediction can be performed for any time point in the future using the same model with the time-dependent arrival and departure rates already estimated. For ANN, there are two ways to predict the occupancy for time $t + k$ at the current time t: 1) applying the next-interval model for $t + 1$ (as shown in Figure 3-13) repeatedly for k times, using predicted values as part of the inputs; and 2) applying a direct model for $t + k$ (see Figure 3-16), which needs to be trained separately beforehand. It is expected that a direct ANN model for $t + k$ would outperform repeated application of the next-interval ANN model. However, since k is not fixed (travelers could query for any time point in the future), training a separate model for each possible k value could be time-consuming.

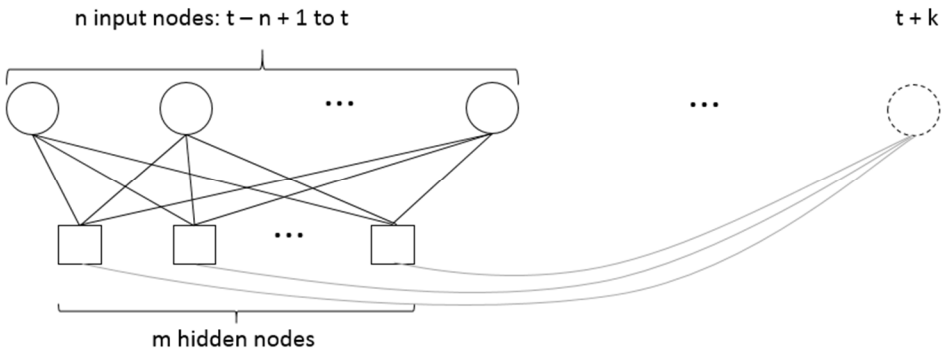

Figure 3-16 Neural Network Structure for

Prediction of Interval $t + k$

Using data from Case Study 3, where the real-time occupancy is collected every 2 minutes, we compare the performance of our method and ANN (repeated application of a next-interval model) for various prediction intervals ranging from 2 to 12 minutes. A direct ANN model to predict for 12 minutes in the future is developed separately and compared against our approach as well.

The resulting MAREs are shown in Table 3-4. When repeatedly applying the next-interval model, the MARE from ANN increases faster than our approach as the time point of interest gets further out in the future. While the next-interval ANN model achieves statistically similar performance compared to our method for 2 and 4 minutes in the future, its MAREs become statistically larger than those from our approach starting from the prediction for 6 minutes in the future. For the prediction for12 minutes in the future, the MARE from repeatedly applying the next-interval ANN is almost twice that of

our approach. On the other hand, the direct ANN model specifically trained to predict for 12 minutes in the future is able to achieve the same performance as our approach.

Table 3-4 MARE for Different Prediction Intervals

Prediction Interval (min)	2	4	6 *	8 *	10 *	12 *
Our Approach	0.489%	0.812%	1.061%	1.283%	1.482%	1.672%
Repeated Application of the Next-Interval ANN	0.486%	0.867%	1.303%	1.803%	2.378%	2.998%
Direct ANN for 12 minutes in the future	--	--	--	--	--	1.697%

* The MAREs from repeated application of the next-interval model and from our approach are statistically different at 99% confidence level.

In terms of computation time, since Case Study 3 has significantly more data than Case Study 1, it is expected that it would take days to tune the ANN using the same genetic algorithm setting in Vlahogianni et al (2015) (50 chromosomes and 100 generations). Instead, we performed the genetic algorithm with 10 chromosomes and 10 generations, and the corresponding tuning time is 1 hour 37 minutes for the next-interval model and about the same for the direct model for 12 minutes in the future. Parameter estimation in our approach took less than 2 minutes.

3.5.3 Prediction without Real-time Update

For prediction without real-time data, we compared our method with the feature-weighted average method in Tamrazian et al. (2015). The resulting MAREs for both case studies 1 and 3 are shown in. Our approach leads to considerably lower MAREs in both cases. A two-sample T-test also confirms that the differences are statistically significant.

Table 3-5 MARE for Prediction without Update Methods

Approach	Case 1	Case 3
Proposed in This Chapter	8.907%	6.363%
Feature-Weighted Average	10.676%	7.680%

3.6 Summary

This chapter proposes a model-based practical framework to predict future parking occupancy from historical occupancy data alone. Being model-based rather than purely statistical, the parameter estimation methods and parking occupancy prediction in the predictive framework benefit from time-dependent analytical properties of the underlying queuing model and are computationally efficient. In addition, several practical considerations for real world implementation are accounted for in the framework, with methods proposed to handle variations of arrival and departure patterns from day to day and within a day, including special events. Using both simulated and real data, we have demonstrated that the proposed framework and methods are able to effectively estimate

the arrival and departure rates offline from historical occupancy data alone, and accurately forecast parking availability with and without real-time occupancy data. It is found that our approach delivers equal or better performance compared to several pure machine-learning methods from recent literature, but requires the computation time that is orders of magnitude less to tune and train the model. Additionally, our approach can predict for any time in the future with one training process, while machine-learning methods have to train a specific model for a different prediction interval to achieve the same level of accuracy.

4 A REINFORCEMENT LEARNING APPROACH FOR USER-OPTIMAL PARKING SEARCHING STRATEGY ON A NETWORK EXPLOITING NETWORK TOPOLOGY

4.1 Introduction

As an essential component of the urban transportation network, parking searching has been widely considered as a significant reason for congestion in downtown areas. According to empirical studies (Van Ommeren, Wentink, and Rietveld 2012; Shoup 2006), cruising for parking is responsible for 30% of traffic (trips) on average in urban areas. In a more recent research, Giuffrè et al. (Giuffrè, Siniscalchi, and Tesoriere 2012) found that cruising for parking results in a peak increase of about 25 – 40% of the traffic flow. Simulation studies incorporating parking search at macroscopic (Cao and Menendez 2015; Geroliminis 2015; Leclercq, Sénécat, and Mariotte 2017) and microscopic level (Benenson, Martens, and Birfir 2008; Levy, Render, and Benenson 2015) also show the degradation of traffic condition caused by cursing for parking. For example, by exploiting the properties of the macroscopic fundamental diagram, Geroliminis (Geroliminis 2015) modeled the dynamics of parking searching and showed the cruising affects all travelers even those with a destination outside the limited parking region.

Recognizing the significance of parking searching problem, parking guidance and information systems, which often involve parking information collection, processing, and distribution, have emerged to help drivers find parking spaces. Parking information

collection is often achieved by sensors at entrances/exits or at individual parking stalls (Idris et al. 2009). Crowdsourcing is also an emerging approach to parking information collection. Furthermore, to provide future parking availability information, many studies have proposed prediction methods with model-based, statistical, or machine learning approaches (e.g., *10*, *11,* and the articles cited therein). Data distribution relies on efficient communications between drivers and the service provider (Geng and Cassandras 2012; Rajabioun, Foster, and Ioannou 2013); and smartphone-based services have become a popular solution. Available mobile services currently on the market include mobile payment, parking information provision, and reservation (e.g., ParkMobile, ParkMe, BestParking etc.).

We argue, however, the effectiveness of providing parking information (current parking availability, predicted future parking availability, and predicted parking search time) alone to users might be limited. The parking information collection methods mentioned previously may be unreliable under abnormally high parking demand, which usually suggests special events and unfamiliar drivers, when effective parking guidance and information services are most needed. Under such circumstances, it is common that parking spaces are reassigned or reserved (special event); and double parking is also common for parking facilities with narrow stalls (unfamiliar drivers). While some researchers found parking information such as occupancy information, parking searching time (Ibeas et al. 2014) and future parking availability (Chaniotakis and Pel 2015) significantly reduce users' cruising time, others showed that the benefit depends on the

level of congestion (Asakura and Kashiwadani 1994). The benefit is limited when the parking capacity is almost saturated.

Providing travelers with parking searching strategies (suggested actions for a user to take in any given state of the parking searching process) could be a more effective approach to parking management, especially for unfamiliar drivers and for special events. The parking searching problem can naturally be viewed as a Markov Decision Process (MDP) where the outcome is random. User-optimal strategies can be formed if the underlying Markov transition probabilities and the cost/reward structure associated with the outcomes are known. To make the MDP formulation more realistic, Tang et al. (Boyles 2014) introduced driver memories in modeling individual driver's parking searching process. Drivers are assumed to have memories of the probability of finding parking at each facility, which is set to either 1 or 0 upon the outcome of a visit to a facility, and are later gradually reset to an *a priori* value. In their approach, the state space would grow exponentially with the memory size. Additionally, the actual probabilities of finding parking (the *a priori* values) are difficult to define and calculate, even with accurate occupancy data. To address this issue, learning algorithms can be introduced to approximate the optimal strategy. Commuters already form user-optimal strategies by learning from their day-to-day experiences. While learning is limited during non-recurring parking searching for an individual driver (e.g., when going to a special event or visiting a special destination) due to limited interaction with the environment, it is possible to introduce learning into parking guidance and information services that can

leverage experiences from numerous users attending the same event or visiting the same destination.

In this chapter, we consider the setting where a central server is employed by a parking guidance and information service provider, and users of the service interact with the central server through a mobile phone app or alike. The parking searching process on a network with uncertain parking availability can naturally be modeled as a Markov Decision Process (MDP). Such an MDP with full information can easily be solved by dynamic programming approaches. However, the probabilities of finding parking are difficult to define and calculate, even with accurate occupancy data. Learning algorithms are suitable for addressing this issue. The central server collects data from numerous travelers' parking search experiences in the same area within a time window, computes approximated optimal parking searching strategy using a learning algorithm and distributes the strategy to travelers. Figure 4-1 describes the users' interactions with the server and the environment. Through the app, the central server gathers information on whether a user successfully finds parking or not either through location data or user reports. It then learns from the parking search experiences from all the users in the same area within a reasonable time window to compute an approximated user-optimal strategy. The approximated user-optimal strategy is distributed to those still searching for parking within the same time window through the mobile app. This learning cycle repeats for each time window. For users whose searches are not completed within a time window / learning cycle, updated recommendations can be pushed to them through the mobile app.

Alternatively, this update can be skipped and the outcomes of these users' searches following old strategies would provide valuable data and jump-start the next learning cycle. The latter might even be preferred by the users, especially when the time windows are small, so that they do not receive constant updates.

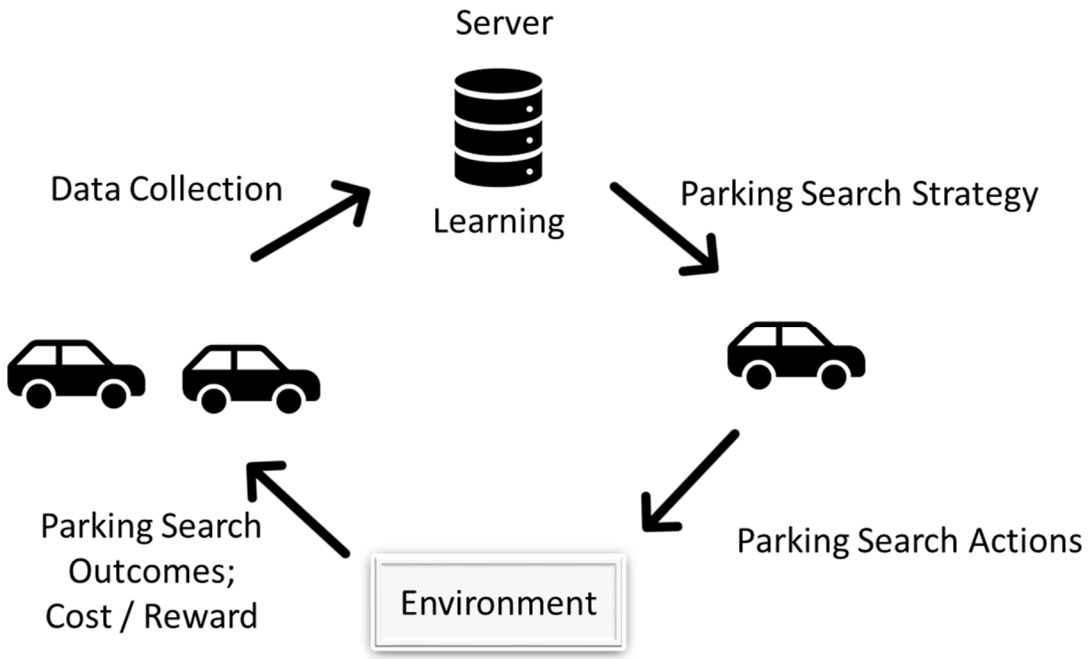

Figure 4-1 Framework

Note that the users are not required to adopt the recommended strategy. Diverse actual behavior policies adopted by the users arguably would help with the convergence of the learning algorithm. While travelers' actual searching behaviors would affect the underlying probabilities of finding parking at each facility, it should also be noted that we are not trying to predict the actual parking probabilities for the current and future time windows. The actual probabilities are highly likely to fluctuate continuously; but the

proposed system and learning algorithm will model the probabilities as constant in a given time window / learning cycle.

Therefore, it is desirable that the parking conditions are relatively stable during each time window / learning cycle (from drivers exploring the environment and collecting data to an estimated user-optimal strategy being distributed to drivers) as shown in Figure 4-1 Otherwise, the knowledge and user-optimal strategy learned during exploration would be applied in a different environment and would not be effective. The time windows could be pre-determined for recurrent special events (such as seasonal sports events) where the demand patterns are generally known. They could also be determined in real time when the central server detects a shift in the parking search outcomes from the users; or if the parking guidance and information service provider is also able to obtain data (such as real-time occupancy and capacity of a parking facility) from infrastructure managers. With data from the supply side, it is possible to estimate the parking probabilities and predict future parking availability. Our previous work (*11*) has proposed a rolling-horizon framework and an iterative approach for this purpose. The framework is also able to detect demand pattern shifts and could potentially be applied for the time window determination. The focus of this chapter, on the other hand, is on reducing the amount of data and time required to train the learning algorithm. This is crucial so that the time window can be sufficiently short, if needed. Additionally, being able to learn the optimal strategy with limited data frees the parking guidance and information service from a high

market penetration requirement. The service and algorithm would not need all or a large percentage of travelers to participate in order to work.

We propose a modified Q-learning algorithm that can be adopted by such parking guidance and information services to provide user-optimal parking search strategies for individual drivers. As a reinforcement learning algorithm, Q-learning is a model-free off-policy method where the learning is performed through agent actions and environment feedback. Since user actions are choosing the parking facility to visit next, and the reward structure is closely coupled with the transportation network through travel cost, it is possible to incorporate and take advantage of the network topology in the learning algorithm. We propose a Q-learning-based algorithm that utilizes the underlying transportation network topology, which can dramatically reduce the size of the state space and improves the convergence speed and the solution quality.

To this end, the contributions of this chapter are: 1) the first to introduce reinforcement learning into parking guidance and information systems to provide user-optimal parking strategies to individuals; 2) a modified reinforcement learning algorithm utilizing the topology of the transportation network, greatly reducing the amount of data required for training.

4.2 Methodology

This section discusses the model and methodology proposed in this chapter. Section 4.2.1 describes our model for the parking searching MDP problem. Section 4.2.2

provides an overview of reinforcement learning methods. Section 4.2.3 explains in detail the proposed learning algorithm that utilizes the transportation network topology.

4.2.1 Modeling Parking Searching Problem as a Markov Decision Process

Consider a strongly connected urban transportation network with a set of parking facilities (nodes), denoted as $N = \{1, 2, \dots, n\}$, where there is a path between every pair of nodes. The probability of finding parking at each facility i is denoted as p_i. This network model can incorporate both off- and on-street parking facilities. For on-street parking, a dummy node can be created with its corresponding probability and inserted in the middle of the modeled street. A driver starts from origin O, tries to find parking in the transportation network, and travels to destination D using other modes (e.g., walking). Without loss of generality, we assume that no parking is allowed at the origin O or the destination D. The goal of a parking strategy is to minimize a driver's expected cost (or maximize a driver's reward) of parking, which includes the cost of cruising to search for parking and the travel cost from a parking facility to her final destination. For the non-recurring parking searching scenarios considered in this dissertation (e.g., when going to a special event or visiting a special destination), p_i is unknown to the users or the parking guidance and information system. While p_i constantly fluctuates in reality, we assume that p_i remains relatively stable in each time window / learning cycle as discussed in Section 4.1; and the learning algorithm will treat p_i as constant.

The parking searching problem can be formulated as an MDP. Each node i in the set N corresponds to two states: i_R and i_P, representing a traveler not finding parking / found

parking at node i respectively. If a traveler is in state i_R, she can take action a_{ij} $(i, j \in N, j \neq i)$, representing the action of driving from node i to node j and searching for parking there. Note that node j may not be the immediate adjacent facility to node i in the physical road network. Each action a_{ij} leads to two possible outcomes (j_R and j_P) depending on the probability of finding parking at node j. If a traveler is in state i_P, the only action allowed next is a_{iD}, traveling to the destination using other modes.

Figure 4-2 and Figure 4-3 illustrate the physical network and the transition graph for a 3-node example. In Figure 4-2, only driving links but no walking paths are shown. Note that the origin node is not directly connected to nodes 2 and 3 (Figure 4-2), but a driver at origin O can take actions $a_{oj}, \forall j \in N$ (Figure 4-3). Action a_{02} means that the driver goes from origin to node 2 to search for parking, bypassing node 1 even it is on his way. A driver in state $the\ 1_R$ (physically at node 1 but did not find parking there) can take action a_{1j} $(j = 2, 3)$, driving to node j and searching for parking there. If he takes action a_{12}, then there is a probability of p_2 that he finds parking and reaches state 2_P; he will then take action a_{2D} to travel to the final destination using other modes (e.g., walking). Otherwise he would end up with state 2_R and continue searching. Also note that states $1_P, 2_P, \ldots n_P$ are in fact terminal states, as the only action allowed from these states is traveling to the destination D using other modes. Given the states and actions above, we are able to define the rewards based on the actions taken. For searching actions, the reward of $a_{ij} (i \in N \cup \{O\}, j \in N, j \neq i)$, is simply defined as $-d(i, j)$, the negative shortest path travel distance between node i and j. Similarly, after a traveler finds

parking, the reward is defined as the negative of the travel cost to the destination by other modes.

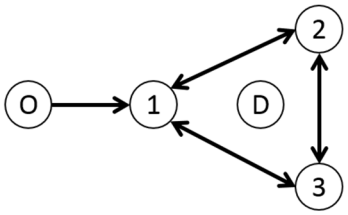

Figure 4-2 Physical Network (3-Node Example)

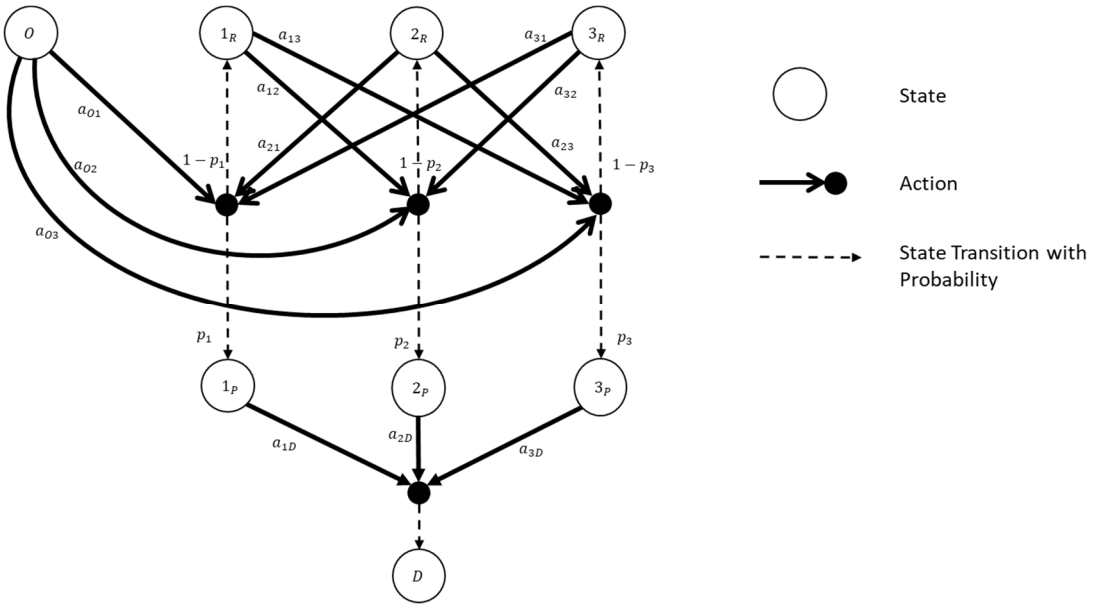

Figure 4-3 Transition Graph (3-Node Example)

The optimal value of a state is the total expected reward staring from this state to the destination state D, following the optimal strategy. If the transition probabilities are known, the problem can be solved exactly using dynamic programming methods such as value iteration and policy iteration. When the transition probabilities are unknown to

agents in the system, reinforcement learning methods can be adopted to estimate the optimal values and generate an approximated optimal strategy based on the values learned. Note that the optimal values of terminal states $s \in \{D, 1_P, 2_P, \dots, n_P\}$ are determined. We are interested in estimating the optimal values of state $s \in \{O, 1_R, 2_R, \dots, n_R\}$.

4.2.2 Overview of Reinforcement Learning

With the advancements in theory, algorithms and computational power, reinforcement learning algorithms have already been effectively applied to transportation problems including signal control (Abdulhai, Pringle, and Karakoulas 2003; Arel et al. 2010; Cai, Wong, and Heydecker 2009; Prashanth L. A. and Shalabh Bhatnagar 2011), train rescheduling (Šemrov et al. 2016), travel behavior (Arentze and Timmermans 2003) and autonomous vehicle control (Dai et al. 2005; Desjardins and Chaib-draa 2011).

In reinforcement learning, an agent learns how to achieve a certain goal by exploring the environment: it takes an action at its current state in the environment, moves to the next state depending on the probabilistic outcome, and receives rewards. The action and reward pairs are used to update the estimated optimal value of each state.

On-policy and off-policy methods are two classes of learning methods. On-policy methods (e.g., SARSA and TD(λ), see (SUTTON 1998)) start with a given policy that adopts the current optimal actions most of the time but also has small probabilities of taking other actions for further exploration, and learn the value of each state under this policy until the optimal values and actions converge. On the other hand, off-policy

methods (e.g., Q-learning (Watkins 1989), R-learning (Schwartz 1993)) can learn the state-action values of a target policy that the agents do not necessarily follow in their exploration.

For our problem, off-policy methods are more appropriate since the central server does not know the actual behavior policies being adopted by travelers in the parking search data collected. Q-learning is a common off-policy algorithm learning directly from the consequence of actions. It can be proven that given sufficient training data under any soft behavior policy, where the probability of taking the current optimal action is less than 1, even purely random behavior policy, the optimal action values learned will converge with probability 1 (SUTTON 1998).

4.2.3 Proposed Learning Algorithm

We propose a modified Q-learning algorithm that takes advantage of the network topology and dramatically reduces the size of state space. The proposed algorithm takes travelers' parking search actions and outcomes (regardless of the actual behavior policies adopted) and generates an estimated optimal parking searching strategy. It does not require a large amount of data for training, and is able to converge quickly. It is suitable to provide an approximated optimal strategy for parking searching problem with limited information.

We will briefly introduce the Q-learning algorithm first, followed by the proposed algorithm which is based on Q-learning.

Q-learning is an off-policy learning approach approximating the underlying value of state-action function $Q(s,a)$ $(s \in S, a \in A_s)$ directly when an agent at state s takes action a and ends up at state s':

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r_a + \gamma \max_{a' \in A_{s'}} Q(s',a')) \qquad (4.1)$$

where α is the learning rate and γ is the discount factor. Since we are trying to minimize the total cost of the parking search process without discounting the cost of any action in our problem, the discount factor $\gamma = 1$. r_a is the immediate reward of action a and $\max_{a' \in A_{s'}} Q(s',a')$ is the maximal possible reward afterwards. Equation (4.1) means that $Q(s,a)$ would be updated as the weighted average of the current value of $Q(s,a)$ and the reward following the current optimal strategy based on current Q values. Note that the Q values can be arbitrarily initialized because in each update, intuitively speaking, we are adding more truth and discounting the effect of initialized value by α. In fact, it can be proven that the algorithm converges as long as all state-action pairs continue to be updated (SUTTON 1998).

We propose a modified Q-learning algorithm for the parking searching problem, taking advantage of the topology of the transportation network. Note that $Q(i_R, a_{ij})$ represents the expected value of taking action a_{ij} from state i_R. This means that $Q(i_R, a_{ij})$ is comprised of two components: the travel cost from current node i to j, and the sum of all the expected following rewards starting from searching at node j. Now consider two state-action pairs (i_R, a_{ik}) and (j_R, a_{jk}) where the actions are driving to the same node k

from different current nodes i and j. The only difference between $Q(i_R, a_{ik})$ and $Q(j_R, a_{jk})$ lies in the first component, their travel costs. The sum of all the expected following rewards starting from searching at node k would be the same, denoted as $V(k_R)$. Define

$$\begin{aligned} V(k_R) &= Q(O, a_{Ok}) - d(O, k) \\ &= Q(i_R, a_{ik}) - d(i, k) = Q(j_R, a_{jk}) - d(j, k), \qquad \forall i, j \in N \end{aligned} \tag{4.2}$$

Since the travel cost is deterministic, we only need to update one value $V(k_R)$ for all the actions going to the same location k when any action a_{ik} is taken. From equation (4.2), updating $V(k_R)$ is essentially updating $Q(i_R, a_{ik}),\ \forall i \in N$ as well as $Q(O, a_{ik})$. In other words, with the original Q-learning equation (4.1), only one Q value is updated when a state-action pair occurs; but with equation (4.2), when a state-action pair occurs, the Q values for all state-action pairs with the same destination node are updated at the same time. This will greatly reduce the number of updates to be performed in the learning process.

Pseudo code:

Initialize $V(s)$, $count(s)$

Repeat (for each parking search trajectory):

    Initialize the starting node i,

    Repeat (for each parking facility j visited in the search trajectory before finally finding parking):

$\alpha(j) \leftarrow \frac{1}{count(j)^{\frac{2}{3}}}$

$V(j_R) \leftarrow (1-\alpha)V(j_R) + \alpha[r_{a_{ij}} + \underset{k \in A_{j_R}}{max}\big(V(k_R) + d(j,k)\big) - d(i,j)]$

count(j) $\leftarrow$ count(j) + 1

$i \leftarrow j$

where $count(j)$ records how many times facility j is visited. The learning rate α is calculated separately for each node j based on $count(j)$. For each node, the series of learning rate must add up to infinity while the summation of the square of α must be finite to ensure convergence (Melo 2001). After estimating V values, the Q values can be derived by reversing equation (4.2). The optimal policy can then be derived by choosing the action with maximal value at each state.

4.2.4 Benchmark Algorithm

To evaluate the proposed algorithm later in our numerical experiments, we choose the value iteration method as a benchmark where all the parking probabilities are assumed known. Denote $V(s)$ as the expected total rewards if a traveler starts from state s and taking optimal actions until he reaches the destination. Value iteration iteratively updates $V(s)$ by taking the optimal action based on the current $V(s)$ estimates:

$$V_{i+1}(s) = \underset{a \in A_s}{max}\{\Sigma_{s'} P_a(s,s')(R_a(s,s') + \gamma V_i(s'))\}$$

In the above equation, $V_i(s)$ is the i^{th} estimation of $V(s)$, $P_a(s,s')$ is the probability of transitioning from state s to s' under action a, and $R_a(s,s')$ is the corresponding reward. $V_0(s), \forall s \in S$ can be arbitrarily initialized. The discount factor γ is 1 because we are

considering the total undiscounted reward. As an undiscounted Markov decision problem, the convergence has not been completely understood; but for our problem, the convergence of value iteration can be proved from a sufficient condition proposed in (Federgruen, Schweitzer, and Tijms 1978). This guarantees the validity of using value iteration as our benchmark. Finally, the optimal state values $V(s)$ can be used to derive an optimal strategy π by:

$$\pi(s) = \arg\max_{a \in A_s} \{\Sigma_{s'} P_a(s, s')(R_a(s, s') + \gamma V_i(s'))\}$$

4.3 Numerical Experiments

Numerical experiments are conducted with randomly generated parking probabilities on a toy network to investigate the performance of the proposed method. The transportation network considered has six nodes. The network topology and associated travel costs as shown in Figure 4-4. The proposed method is compared with the original Q-learning, a greedy nearest-node strategy and the optimal strategy calculated using value iteration from the exact probabilities of finding parking. The experiments show that the proposed learning algorithm could generate a searching strategy close enough to the optimal strategy with a reasonable size of training data. Additionally, sensitivity analysis is performed regarding the amount of data needed.

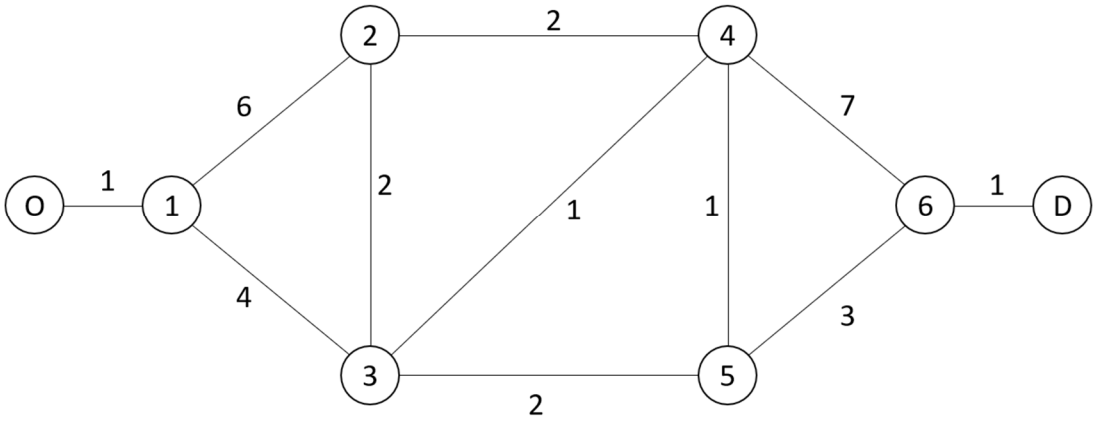

Figure 4-4 Six-Node Network

4.3.1 Methods Comparison

The comparison among methods is performed under 1000 different scenarios. In each scenario, the actual probabilities of *not* finding parking are randomly generated between [0.5, 1] for each parking facility, simulating a relatively congested parking condition. In each scenario, 100 parking search trajectories are generated from a purely random search policy as the training data used for learning. This is a reasonable data size considering the scale of a real special event.

We first compare the ranking (in terms of expected cost) of the strategies obtained by the three approximate methods: the greedy nearest-node strategy, the Q-learning method, and the proposed modified Q-learning method. From Table 4-1 Cost Ranking Counts, it can be seen that the proposed modified Q-learning is ranked first (lowest expected cost) in 529 out of the 1000 scenarios; and is ranked last (highest expected cost) in only 99 scenarios. The ranking performances of the original Q-learning and the nearest-node

strategy are similar, both inferior to the proposed modified Q-learning algorithm. Both of them are ranked first for only 200+ scenarios, but last for 400+ scenarios.

Table 4-1 Cost Ranking Counts

Strategy / Rank	Nearest	Q-learning	Revised Q-learning
First	228	243	529
Second	334	294	372
Third	438	463	99

We further benchmark the three approximate methods against the true optimal strategy solved using value iteration. Two performance metrics are examined: the average expected travel cost over the 1000 scenarios, and the performance profile (Dolan and Moré 2002).

In terms of the average expected travel cost over the 1000 scenarios, the true optimal strategy scores 11.56. The average expected cost of the nearest strategy, Q-learning and modified Q-learning are 12.86, 12.92 and 11.77, respectively. The gap between the strategy obtained from the modified Q-learning and the optimal strategy is very small.

The performance profile provides a more detailed, graphical comparison of different algorithms over the same problem set (Dolan and Moré 2002). In this study, each

scenario is a unique problem in the problem set. For each scenario c and algorithm m, $t_{c,m}$ is defined as the expected cost of the resulting strategy. We use the true optimal strategy as a baseline, which is always better or equal to other strategies but unknown to travelers. The performance ratio of algorithm m in scenario c is defined as:

$$r_{c,m} = \frac{t_{c,m}}{t_{c,optimal}}$$

We are interested in obtaining an overall assessment of the performance of the algorithms instead of any particular scenario. Now define

$$\rho_m(\tau) = \frac{n_c}{|C|}, c \in C: r_{c,m} \leq \tau$$

where n_c is the number of scenarios in which the algorithm m has a performance ratio within a factor τ of the best possible strategy. As defined, $\rho_m(\tau)$ is essentially the cumulative distribution function of the performance ratio of algorithm m. For example, a point $(1.05, 0.93)$ on the dotted line in Figure 4-5 indicates that the modified Q-learning algorithm can solve 93% problems in the problem set with a cost smaller than or equal to 1.05 times what the optimal solution takes.

It can be observed from Figure 4-5 that for the easier 40% of the 1000 scenarios, all three approximate algorithms can lead to strategies that have almost the same expected cost as the true optimal strategy (all three curves are almost vertical in the box defined by $\tau \in [1,1.1]$ and $\rho(\tau) \in [0, 0.4]$). On the other hand, for all of the 1000 scenarios, the proposed modified Q-learning would result in strategies that cost no larger than 2 times what the true optimal strategy costs (the $\rho(\tau)$ value reaches 1 before τ gets greater than 2

for the dotted line). But the other two approximate algorithms could lead to strategies that cost much more compared to the true optimal strategy. In the worst case, the estimated optimal strategies obtained from Q-learning could take up to 7.7 times the true optimal cost (the long tail of the solid curve hits $\tau = 7.7$ when $\rho(\tau) = 1$). This might be because the training data set in this case study is limited to 100 search trajectories. The values of the original Q-functions are not properly trained with limited data. In the next section, we will explore the sensitivity of the Q-learning and the proposed limited Q-learning algorithms.

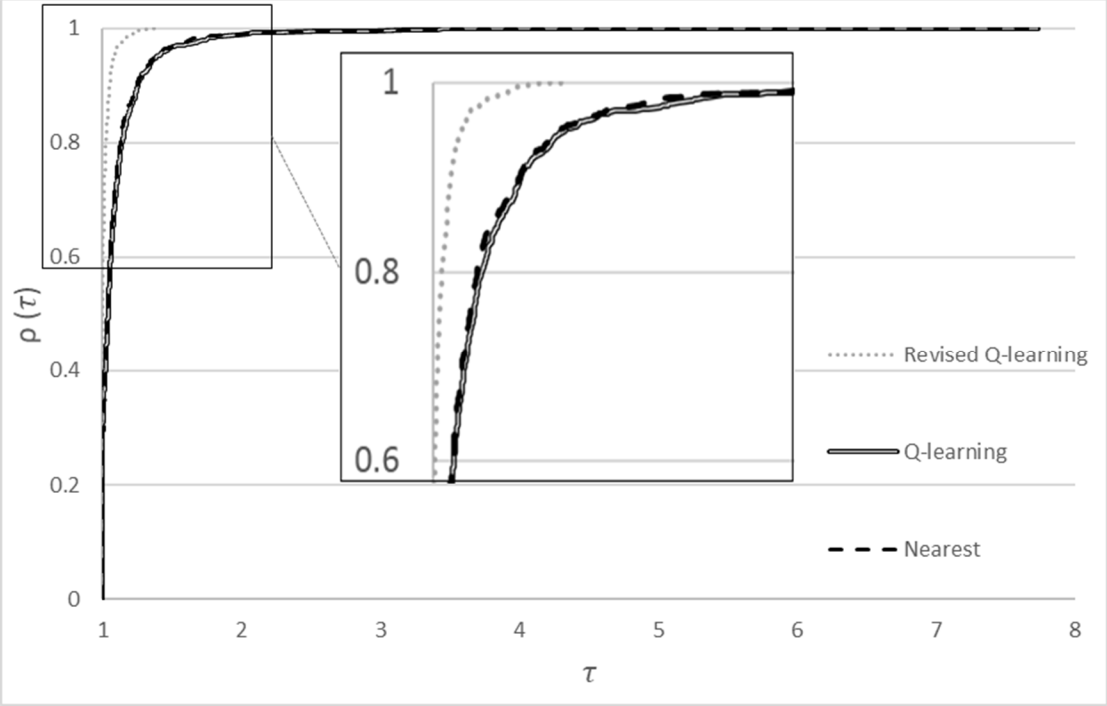

Figure 4-5 Performance Profile of Case Study

4.3.2 Sensitivity Analysis

Applying the same settings as in section 4.3.1, the purpose of this experiment is to investigate how the size of training data would affect the performance of Q-learning and revised Q-learning. The performance measure here is the relative excessive cost (REC), where the optimal cost from value iteration is used as a benchmark.

Table 4-2 Relative Excessive Cost

DATA SIZE	**RELATIVE EXCESSIVE COST**	
(Number of Random Search Trajectories)	Q-Learning	Modified Q-learning
10	17.23%	12.46%
50	15.36%	3.20%
100	11.72%	1.83%
200	6.62%	1.75%
500	2.14%	1.12%
1000	0.71%	0.70%
10000	0.22%	0.28%

From Table 4-2 Relative Excessive Cost, it can be seen that as the size of the training data set increases, the performance of the Q-learning gradually catches up and later surpassed modified Q-learning when more than 1000 search trajectories are involved.

However, the parking conditions could have changed by the time enough data is collected and an estimated user-optimal strategy is pushed to the drivers. The proposed modified Q-learning, on the other hand, is able to generate searching strategies close enough to the true optimal strategy with as few as 50 search trajectories.

4.4 Summary

This chapter investigates the idea of introducing learning algorithms into parking guidance and information systems that employ a central server, in order to provide estimated user-optimal parking searching strategies to individual travelers. The central server collects data from numerous travelers' parking search experiences in the same area within a reasonable time window, computes approximated user-optimal parking searching strategy, and distributes the resulting strategy to travelers who are still searching in the same time window. This cycle repeats for each time window. A modified Q-learning approach is proposed. The proposed learning algorithm takes advantage of the network topology and dramatically reduces the size of the problem. Our numerical experiments have demonstrated that the proposed algorithm is able to produce high quality solutions with limited training data, and thus has great potential to be applied in real time.

5 A PLANNED SPECIAL EVNET NETWORK OPTIMIZATION MODEL WITH PAKRING AND RIDESHAIRNG

5.1 Introduction

The previous two chapters focus on reducing parking cruising for travelers without reserved parking spaces. In this chapter, additional components of the transportation system would be taken into consideration. We optimize the PSE network from a planner's perspective, providing route recommendations to travelers with and without reserved parking spaces as well as ridesharing vehicles. Lane configurations (lane reversal, turning movement assignment) also play an important role at the planning stage. A planned special event such as a sports game or a concert at the heart of an urban area can greatly affect the regular operation of a transportation system due to significantly increased travel and parking demand that often leads to severe congestion. For event-goers, getting to the event location and finding parking could be a struggle. This has led to increasing public demand for traffic and parking information and services during PSEs. Some attendees are turning to ride-sharing services such as Uber/Lyft, rather than driving by themselves. However, the allocation of ridesharing lanes/drop-off locations needs to be carefully planned. Otherwise, ridesharing may create additional issues such as constant interruption to traffic because of intensive lane changing behavior around the drop-off locations and pedestrian-vehicle traffic conflict. For event planners and traffic operators, the massive congestion is a concern for both traffic operations and traffic safety. To facilitate traffic, the road network is usually reconfigured for a PSE, such as

road closures, reversed lanes, and limited access to parking facilities. For recurring PSEs, event-goers are often provided with recommended routes to designated parking areas in advance. Such network reconfiguration and route and parking recommendations are, however, often ad-hoc in practice; and the effectiveness of such measures is often unclear. Related research on congestion mitigation during the ingress of a PSE is limited, we expand our literature review to include some additional work on optimizing network configurations.

A few studies (Latoski et al. 2003; Silvers 2012) have summarized all the aspects to consider during a PSE by high-level descriptions. By designing efficient transit systems (Ruan et al. 2016), travelers are encouraged to take public transportation thus reducing the vehicular demand entering the PSE network. Existing network optimization problems with bi-level structure adopt various algorithms. Wong and Wong (Wong and Wong 2003) applied branch & bound to assign turning movements to lanes for an isolated junction. It works because the feasible region in one single junction is relatively small. For optimization of the entire network, customized heuristics have been adopted to solve the problem. The genetic algorithm is used by Goerigk et al. and Meng et al. (Goerigk, Deghdak, and Hebler 2014; Meng, Khoo, and Cheu 2007) with feasibility restoration and customized mutation. Teydes and Ziliaskopoulos (Tuydes and Ziliaskopoulos 2007) solve the problem with Tabu-based heuristics. Xie and Turnquist (Xie and Turnquist 2011) also applied Tabu-search approach but together with Lagrangian relaxation. A bottleneck relief heuristic is developed in (Kim, Shekhar, and Min 2008). In each iteration, the

algorithm finds a min-cut of the current network and flip lanes across the min-cut until max-flow is not increasing.

In this chapter, we propose a bi-level model combing optimization and simulation approaches. The upper level is an optimization model aiming to minimize total travel time experienced by travelers. The decision variables are the number of lanes on each link, route recommendations, and ridesharing drop-off locations. In the lower level, a link transmission model (LTM) incorporating parking and ridesharing is used to evaluate the total travel time and provide feedback to the upper level. We also developed effective and efficient heuristic solution algorithms for the PSE traffic planning problem. A case study of Super Bowl XLIX held in Glendale, AZ in 2015 was conducted. The results show that our methods and approaches are able to produce an effective comprehensive traffic plan with reasonable computation time. The main contributions of this work are 1) explicitly modeling the traffic dynamics of parking searching and ridesharing on a network during a special event; and 2) a solution algorithm to generate a comprehensive PSE traffic management plan.

5.2 Network Optimization and Traffic Flow Modeling

The PSE traffic planning method proposed in this paper is a bi-level model. The traffic planner on the upper level tries to minimize the total travel time by changing the lane reversal settings, route recommendations and drop-off locations. The upper level problem is formulated with linear constraints induced by the physical feasibility of network design. As for the lower level, with all the components interacting with each other, it is extremely

difficult to obtain a closed-form representation. Instead, a link transmission model (LTM) based simulation of the ingress traffic dynamics is performed to evaluate the performance of a given network design. The simulation includes parking search behavior by modeling the probabilities of finding parking at destination nodes. Ridesharing is also incorporated considering the reduced capacity and congestion at drop-off locations. The planner can get feedback from the simulation results such as bottleneck locations and link/route travel times. Improvements can then be proposed and analyzed correspondingly.

The rest of this section is organized as follows. Section 5.2.1 explains the upper level optimization model. Details about traffic dynamics will be introduced in section 5.2.2. In section 5.3, two algorithms are proposed to solve the bi-level model. A straightforward genetic algorithm with link expansion is proposed as a benchmark algorithm, it can only handle small networks. While the congestion relief algorithm in section 5.3.2 iteratively improves lane configurations for each pair of ridesharing location and route recommendations.

5.2.1 Optimization Model

In this section, we first introduce how a physical road segment is transformed into links to be used in the model. The objective function, decision variables and constraints of the optimization model are then explained in detail.

5.2.1.1 Network Model

Each physical road segment is modeled as three separate links: one main block area in the middle where only through movement is allowed, and two transition areas on both ends

of the road segment that have the ability to model additional turning bays. Links in transition areas are where sending and receiving traffic flow from other main block links take place.

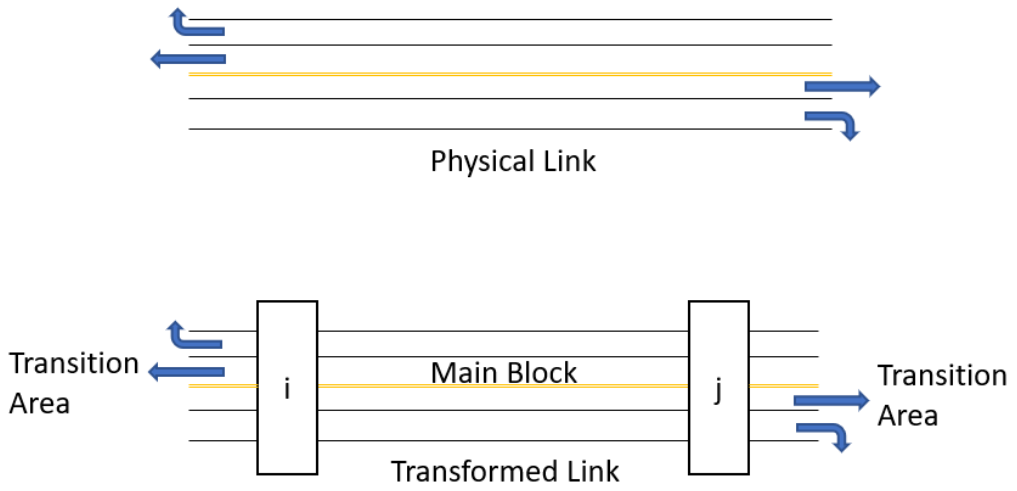

Figure 5-1 Link Model

As shown in Figure 5-1, node i and node j are added to the physical road segment. Link (i, j) is the main block link from node i to node j; and link (j, i) is for the opposite direction. The transition areas connect this main block to other main blocks. The transition area attached to node i contains all the incoming links to node i and outgoing links from node i. It is worth mentioning that for simplicity, we only consider exclusive turning lanes in this paper. Even though shared turning lanes can be easily added, further assumptions on travelers' behaviors would be required in order to decide whether to assign travelers to exclusive turning lanes or shared turning lanes.

5.2.1.2 Sets and Parameters

Notations for sets and parameters to be used in this Chapter are listed below:

Table 5-1 Sets and Parameters

$t \in T$	Time
o ∈ O	Set of origins
$w \in W$	Set of OD pairs
$p \in P^w$	Set of paths for OD pair w
N	Set of nodes
L	Set of links
$L_M \subset L$	Links in main blocks
$L_T \subset L$	Links in transition areas
$S_{ij}, (i,j) \in L_M$	Maximum number of lanes on main block link (i,j)
$S_i, i \in N$	Maximum number of lanes in transition area i
d_t^w	Travel demand of OD pair w at time t

Note that since lane reversal is allowed, we set $S_{ij} = S_{ji}$ as the total number of lanes in both directions. Similarly, S_i is the maximum number of lanes in the transition area attached to node i.

5.2.1.3 Decision Variables

Three sets of decision variables are considered in the model. Some intermediate variables are also listed below.

$x_{ij}, (i,j) \in L_M$	Ridesharing drop-off link (binary)
$u_{ij}, (i,j) \in L$	Number of lanes from node i to node j (integer)
$a_{ij}^{w}, w \in W$ & $(i,j) \in L$	Link (i,j) is on the recommended path or not for OD w (binary)

Intermediate variables:

f_{ijt}^{p}	Number of private vehicles on link (i,j) at time t
$cnv_{ijt}^{up,p}$, $cnv_{ijt}^{down,p}$	Cumulative number of vehicles on upstream & downstream ends of link (i,j) at time t on route p

5.2.1.4 Objective Function and Constraints

Our network optimization model is formulated as follows

$$\underset{(\boldsymbol{x},\boldsymbol{a},\boldsymbol{u})}{\text{Min}} \quad \sum_{t} \sum_{(i,j)\in L} \sum_{p} (f_{ijt}^{p})$$

Subject to:

$$\boldsymbol{cnv^{up}}, \boldsymbol{cnv^{down}} = simulator(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{a}; \boldsymbol{d}) \tag{5.1}$$

$$\boldsymbol{f} = \boldsymbol{cnv^{up}} - \boldsymbol{cnv^{down}} \tag{5.2}$$

$$u_{ij} + u_{ji} \leq S_{ij} \qquad \forall\, (i,j) \in L_M \tag{5.3}$$

$$u_{li} + \sum_{k:(i,k)\in L_T} u_{ik} \leq S_i \qquad \forall\, i \in N, (l,i) \in L_T \tag{5.4}$$

$$\begin{cases} \dfrac{\sum_{k:(k,i)\in L_T} u_{ki}}{S_i} \leq u_{ij} \leq S_{ij} \times \displaystyle\sum_{k:(k,i)\in L_T} u_{ki} \\ \qquad\qquad\qquad\qquad\qquad\qquad \forall\, (i,j) \in L_M \\ \dfrac{\sum_{k:(j,k)\in L_T} u_{jk}}{S_j} \leq u_{ij} \leq S_{ij} \times \displaystyle\sum_{k:(j,k)\in L_T} u_{jk} \end{cases} \tag{5.5}$$

$$1 \leq \sum_{(i,j)\in L_M} x_{ij} \leq K \tag{5.6}$$

$$x_{ij} \leq u_{ij} \qquad \forall\, (i,j) \in L_M \tag{5.7}$$

$$\begin{cases} \sum_{(i,j):o(w)=i} a_{ij}^{w} = 1, \sum_{(i,j):d(w)=j} a_{ij}^{w} = 1 \\ \sum_{j} a_{ij}^{w} = \sum_{j} a_{ji}^{w}, \quad \forall\, i \in N/\{o(w), d(w)\} \\ a_{ij}^{w} \le u_{ij} \end{cases} \quad \forall\, w \in W \qquad (5.8)$$

The objective function of this optimization model is to minimize the total travel time for all travelers. Constraint (5.1) represents a traffic simulator with the network design and time-dependent OD demands as inputs. The outputs are the cumulative numbers of vehicles at the up- and down-stream nodes of each link. More details on the simulator are provided in section 5.2.2. Constraint (5.2) calculates the number of vehicles on each link at any time for any path, which is used in the objective function to evaluate a given design. Constraints (5.3) to (5.5) describe the physical feasibility of the network design. Constraint (5.3) makes sure that the total number of lanes from both directions in the same main block must be smaller or equal to S_{ij}, thus allowing lane reversal in the network design. Similarly, for a transition area i, constraint (5.4) guarantees the sum of outgoing lanes and incoming lanes is less than S_i. Note that the constraint considers outgoing lanes to all downstream links; but only includes one incoming link at a time. This is because incoming traffic is phase-separated for a signalized intersection. The number of receiving links in the transition area i for a main-block link (i,j) is then naturally $\max_{(l,i)\in L_T} \{u_{li}\}$ as a result. Constraint (5.5) ensures the connectivity of the main-block link (i,j). It requires the number of receiving lanes, number of main-block lanes and number of outgoing lanes for one direction to either be all zero at the same time (the

direction is completely eliminated, i.e., full reversal), or all positive at the same time (so that traffic entering the road segment in this direction can move downstream). Constraint (5.6) caps the total number of ridesharing drop-off locations; and constraint (5.7) limits drop-off locations to links with positive number of lanes. Constraint (5.8) guarantees that each OD pair is connected for a given network design by creating a recommended route for each OD pair utilizing links with positive number of lanes.

Other than the general constraints described above, this optimization model can be easily extended to account for realistic considerations. Here we briefly discuss three cases that were encountered in the real-world case study: channelized turning movements, divided main block links and divided transition areas.

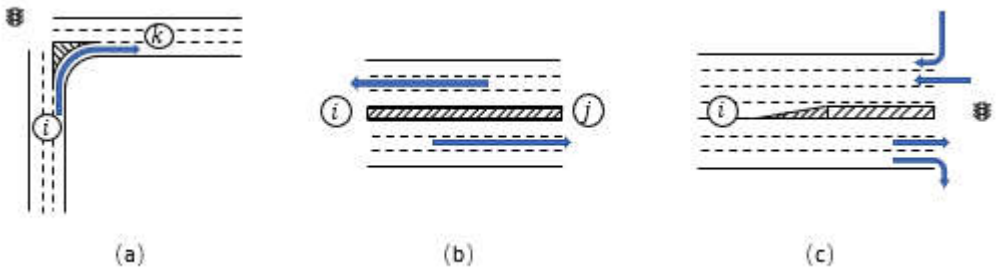

Figure 5-2 (a) Channelized Turning Movements (b) Divided Main Block (c) Divided Transition Area

In addition to channelized right-turn lanes from one surface street to another, highway ramps are another example of channelized turning movements. Lanes designed for channelized turning movements cannot be assigned to other directions. Moreover, other lanes in the transition area cannot be assigned to the channelized direction either. In this

case, the number of lanes for the channelized turning movement is fixed as constant by using a new constraint:

$$u_{ik} = C_{ik} \qquad \forall (i,k) \in L_{T_d} \qquad (5.9\text{ a})$$

Where u_{ik} is the number of lanes on transition link (i,k) and C_{ik} is the number of lanes designed to connect node i and k. L_{T_d} is the set of transition links with channelized turning.

Similarly, for a divided main block as shown in Figure 5-2 (a) Channelized Turning Movements (b) Divided Main Block (c) Divided Transition Area, lanes on both directions can not be changed to the other direction. While it is still possible to reverse the lanes on either side of the barrier by using, for example, traffic cones. Along with this argument the barriers dividing main blocks can be ignored. But in the following of this dissertation we are modeling the barriers explicitly.

$$u_{ij} = C_{ij} \qquad \forall (i,j) \in L_{M_d} \qquad (5.9\text{ b})$$

where u_{ij} is the number of lanes on main block link (i,j) and C_{ij} is the number of lanes on the same side of the barrier. L_{M_d} is the set of divided main block links. It should be noted that $C_{ij} + C_{ji} = S_{ij}$. Here u_{ij} can take any integer value smaller than or equal to C_{ij},

but fixing it as C_{ij} would at least has the same performance as other numbers. It also reduces the number of decision variables.

Figure 5-2 (a) Channelized Turning Movements (b) Divided Main Block (c) Divided Transition Area shows another scenario where a transition area is divided by a physical barrier. The number of incoming lanes and outgoing lanes is fixed in this case. To model this, we introduce a new parameter I_i as the number of incoming lanes in a divided transition area i. Then:

$$\sum_{k:(i,k)\in L_T} u_{ik} = S_i - I_i \qquad \forall\, i \in N \tag{5.9 c}$$

$$u_{li} \le I_i \qquad \forall\, i \in N, (l,i) \in L_T$$

Constraints (5.9 c) should be added if transition area i is divided. And the constraints regarding i in (5.4) can be eliminated because (5.9 c) are stronger constraints.

5.2.2 Traffic Dynamics

The traffic dynamics model implemented in our simulator is based on LTM. This section starts by briefly introducing LTM and the advantages of applying LTM together with the transformed network described in section 5.2.1.1. Sections 5.2.2.2 and 5.2.2.3 explain how parking and ridesharing are incorporated into the basic LTM model. Section 5.2.2.4 presents a comprehensive description of traffic dynamics algorithm.

5.2.2.1 Link Transmission Model

LTM propagates traffic on links based on kinematic wave theory, which allows substantial realism in the representation of queue-propagation (spill back and spill over) and queue-dissipation (Yperman 2007). It keeps track of the cumulative number of vehicles observed at both ends of the links. The flow-density relationship is approximated by a triangular fundamental diagram. At each node, sending flow from incoming links and receiving flow of outgoing links are first calculated. Sending flow on a link l at time t is defined as the outgoing flow from link l between t and $t + \Delta t$ if flow capacity of downstream links is infinite. The sending flow $SF_{l,t}$ is restricted by the link's upstream boundary condition if the link is in a free-flow traffic state. A free-flow traffic state at the upstream boundary will travel forward (downstream) with free-flow speed v_l, and reach the downstream boundary $\frac{D_l}{v_l}$ time units later, where D_l is the length of link l. The maximum number of vehicles that can be sent by link l during Δt, in this case, is restricted to:

$$SF_{l,t} \leq cnv_l^{down}(t + \Delta t) - cnv_l^{down}(t) \leq cnv_l^{up}\left(t + \Delta t - \frac{D_l}{v_l}\right) - cnv_l^{down}(t)$$

Sending flow is also restricted by the flow capacity q_M:

$$SF_{l,t} \leq q_M \times \Delta t$$

Thus, sending flow equals to the more restrictive of these two conditions:

$$SF_{l,t} = \min\left\{cnv_l^{up}\left(t + \Delta t - \frac{D_l}{v_f}\right) - cnv_l^{down}(t), q_M \times \Delta t\right\}$$

Similarly, the receiving flow is restricted by the downstream boundary condition if the link is in a congested condition. The downstream boundary condition will travel upstream with a backward wave speed v_l^b. Therefore, we have:

$$F_{l,t} = \min \left\{ \left(cnv_l^{down}\left(t + \Delta t - \frac{D_l}{v_l^b}\right) + k_{jam}D_l - cnv_l^{up}(t) \right), q_M \Delta t \right\}$$

Node models would then determine how much of the sending and receiving flows can take place. Different node models are applied for origins/destinations, merging nodes and diverging nodes. For more details about LTM, please refer to (Yperman 2007).

Together with the transformed network which separates main blocks and turning areas as described in 2.1.1, this LTM can model the turning bay saturation, starvation, queue, etc. To elaborate, the receiving flow $RF_{l,t} \leq cnv_l^{down}\left(t + \Delta t - \frac{D_l}{v_l^b}\right) + k_{jam}D_l - cnv_l^{up}(t)$. Since Δt is small enough ($\Delta t - \frac{D_l}{v_l^b} < 0$), the first term on the right-hand side $cnv_l^{down}\left(t + \Delta t - \frac{D_l}{v_l^b}\right) \leq cnv_l^{down}(t)$. So the original inequality becomes $RF_{l,t} \leq k_{jam}D_l + cnv_l^{down}(t) - cnv_l^{up}(t)$. This equation indicates that the receiving flow at time t is smaller than the total capacity on link l minus number of current vehicles on link l.

5.2.2.2 Route Assignment Heuristics

At the earliest stage, transportation planners use all-or-nothing to assign traffic to different routes which ignores the “feedback” phenome caused by congestion. Current practice of traffic analysis for transportation planning purposes is based on the concept of

user equilibrium, where a fixed OD matrix is assigned to a network and the travel times on each link can be defined as a function of aggregative flow. The traffic flow under equilibrium (UE) conditions follows Wardrop's principle that all routes used have equal and minimum travel costs. The equilibrium assignment implies strong assumptions that drivers have full information and are fully rational. Clearly both of the above traffic assignment rules are an over-simplification of reality. Generally, people would not be limited to one route during a special event if alternative routes exist. On the other hand, since the traffic state or the network structure could be different from normal conditions, people would not have full information. The traffic assignment heuristic we would like to use should lie between all-or-nothing assignment and equilibrium assignment. Three possible methods to model traffic distribution among possible routes are described in this section: ϵ-soft assignment, deviation assignment and early termination of UE assignment. ϵ-soft assignment assumes that most ($1-\epsilon$ percent) of the demand will take the recommended route and the remaining is equally split among alternative routes. The ϵ value quantifies the proportion of travelers unwilling or fail to follow the recommended routes. This is also the assignment rule adopted in the case studies. Another route assignment logic is assigning demand to routes based on their deviation metric. The deviation is a metric to quantify the difference between a route and the shortest route, which can be customized. For example, it can be defined as the number of links on the current route that are not on the shortest route. The last assignment heuristic discussed is an early termination of the UE assignment. Solving the user equilibrium assignment

using Frank-Wolfe algorithm is mimicking the day-to-day learning process: The traffic flow starts from all-or-nothing in the first iteration, gradually getting more information on the network and ends up with UE when travelers have full information of the network. An early termination of the algorithm generates a flow where travelers have some but not full information.

The exogenous OD demand would be assigned to different routes with selected route assignment heuristic. Start from here, the simulation can run and other components (parking and ridesharing) can be incorporated.

5.2.2.3 Parking Search

This section describes how parking search is incorporated into the LTM simulation. Parking facilities are modeled as nodes in the network. For each parking node n, a probabilistic availability $prob_n$ is attached. In each iteration, the total number of vehicles arriving at n during t to $t + \Delta t$ can be calculated from the cumulative number of vehicles. Assuming no vehicles leave during ingress, the parking probability $prob_n$ is then updated by total available spaces over total arrival (with upper bound 1) at time t. The number of vehicles that can find parking during $[t, t + \Delta t]$ at parking node n is then calculated by $min\,(prob_n \times total\,arrival,\ total\,available)$. This calculation is followed by updating the number of available spaces at the parking facility correspondingly. Those who do not find parking will start a new OD, originating from node n to another parking node. The selection of the next parking node can follow parking search strategies from

Chapter 4. It can be a greedy approach such as always go to the nearest parking lot or the estimated optimal searching strategy with the help of a centralized server.

The probabilities of finding parking at parking nodes usually keep decreasing during ingress. A more realistic parking probability updating mechanism can be adopted for more accuracy. For example, parking facilities that are not specifically set up for the special event might have positive departure rates during ingress. In this case, $prob_n$ can be updated using the queueing model from Chapter 3. $prob_n$ depends on the stochastic departure process which follows Poisson distribution. However, more accuracy usually means more complexity in the simulation. The tradeoff between accuracy and efficiency needs to be carefully tuned. The parking supply in the immediate vicinity for a special event is often not adequate for all incoming vehicles. In this case, vehicles that could not find parking after several attempts are assumed to leave the network (immediate vicinity of event location) and will find parking spots further away from the event location.

5.2.2.4 Ridesharing

Ridesharing service cannot be neglected as an emerging travel mode in special event networks. Three aspects of ridesharing affecting the traffic in the network are modeled in the simulation: 1) the reduced capacity of a link if a certain area of a certain lane is designated as ridesharing drop-off areas; 2) The drop-off duration t_d. We assume the drop-off duration t_d is a random variable following a predefined distribution. We further assume that on the ridesharing link, first-in first-out (FIFO) holds separately among private vehicles and ridesharing vehicles, but not between private and ridesharing

vehicles; 3) The congestion caused by intensive lane-changing behaviors on ridesharing link. Intensive lane-changing along the ridesharing link could reduce the capacity and jam density of the link. To quantify the effect, Jin (Jin 2010) introduced lane-changing intensity ϵ, a parameter describing lane-changing behavior on the macroscopic level that can be incorporated to kinematic wave theory. The idea is to adjust the jam density by doubling the number of lane-changing vehicles in the model, because they are taking two lanes during lane-changing. The lane-changing intensity parameter ϵ is defined as $\epsilon = \frac{m_{lc}t_{lc}}{MT}$ where m_{lc} is the number of lane-changing events; t_{lc} the average time of lane-changing events; M is the total number of vehicles on this link; and T the travel time of the link. The jam density will be divided by $1+\epsilon$, to $k'_{jam} = \frac{k_{jam}}{1+\epsilon}$ (consequently the flow capacity is reduced by the same factor), to account for the congestion caused by lane changing. The new fundamental diagram is shown in the Figure 5-3.

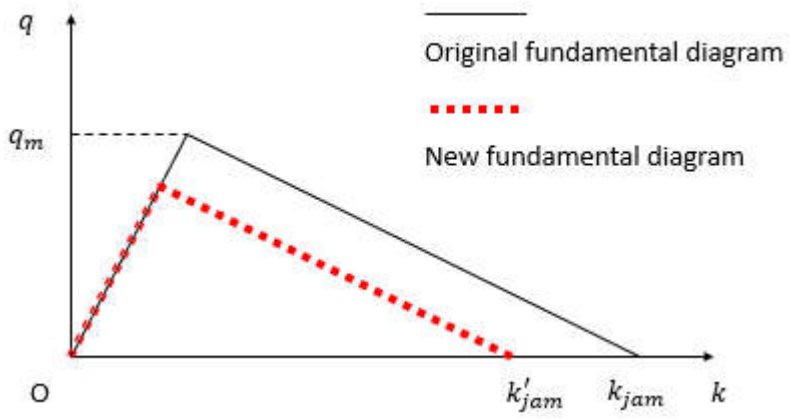

Figure 5-3 Triangular Fundamental Diagram with Lane Changing

5.2.2.5 Traffic Dynamics Algorithm

Build transformed special event network based on decision variables $\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{a}$

For each time interval Δt:

Step 1:

Assign OD demand at time t (d_t^w) to routes based on route assignment heuristic

Step 2:

For all the origin nodes, load route flows $cnv_l^{up}(t + \Delta t) = cnv_l^{up}(t) + d_t^p$, where l is the first link on route p.

Step 3:

For each node (excluding origins and destinations) i link (i, j) is on the main block.

Use node i as merging node, multiple incoming transition links (k, i) are merging into the main block link (i, j)

Use node i as a diverging node, main block link (j, i) diverging into multiple transition links (i, k)

Sending flow, receiving flow and transition flow are first calculated for each node based on cnv^{up} and cnv^{down}. Then total transition flow is divided into different paths based on Link Transmission Model.

Step 4:

For all destination nodes, absorb traffic flow that has reached their destinations.

Step 5:

For all parking nodes, calculate the number of vehicles that have found parking during $(t, t + \Delta t)$. Update parking probabilities. Generate new OD demand $d^{w}_{t+\Delta \mathrm{t}}$ for vehicles that did not find parking.

Step 6:

For all ridesharing links, update the number of vehicles entering and leaving drop-off areas. Update lane-changing intensity $\epsilon_{l,t+\Delta t}$.

5.3 Solution Algorithm

The sets of integer decision variables (route recommendation and sharing locations) and the lower level simulation make this problem impossible to be solved by gradient-based methods. Furthermore, the neighborhood of a feasible solution is combinatorial by nature. Metaheuristics have been widely used in related work and are are good candidates for solving the PSE network optimization model. General metaheuristics make few assumptions about the optimization problem itself and can be applied to a variety of problems. Starting from a set of feasible solutions, a general heuristic procedure tries to iteratively improve the current solution by searching for better alternatives in the

neighborhood. However, general metaheuristic methods could be slow because they do not understand the underlying traffic flow in the network and the neighborhood of a solution is combinatorial in the case of the PSE network optimization model.

Two algorithms are introduced in the following two subsections. The first is a customized genetic algorithm with link expansion (GALE). It is a straightforward application of genetic algorithm with feasibility restoration. After feasibility restoration, an integer programming is formulated and solved to expand the number of lanes on all the links as much as possible. In section 5.3.2, we propose a genetic algorithm with congestion relief algorithm to solve the problem. The solution space is decomposed into two parts: one part consists of ridesharing location and route recommendations and the other part concerns the lane configurations. The first part is still explored by genetic algorithm. But for the second part, the lane configurations are optimized by iteratively estimating the "shadow cost" of increasing and decreasing one lane on a link and adjusting the lane configurations correspondingly.

5.3.1 Genetic Algorithm with Link Expansion

We first adopt a genetic algorithm with problem-specific treatments. It is worth noting that all the decision variables are integer (or binary) in this problem. First, an initial population is generated where the decision variables are randomly assigned values between their lower and upper bounds. Note that the candidates in this initial population might not be feasible. To ensure the candidates are all feasible, an integer programming is then formulated for each candidate to restore its feasibility with minimum alteration. The

objective function is to minimize the total number of lanes changed in order to satisfy the feasibility constraints from the upper level optimization model. With the feasible population, we further increase the value of all u_{ij} variables as much as possible for each candidate in order to make more constraints in (5.3) and (5.4) binding. Intuitively, this treatment means that all available lanes will be assigned to some movement. The candidates will at least have the same performance after this treatment. This is done by formulating another integer programming aiming to reduce the gap between the left-hand side and right-hand side of constraints (3) and (4) in section 5.2.1.4. Suppose $\boldsymbol{u}$ is the current lane configuration and $\boldsymbol{Au} \leq \boldsymbol{b}$ are constraints in the optimization model regarding lane configurations. Then the formulation is to find decision variables $\boldsymbol{\Delta u}$ such that:

Lane Expanding Model (LEM)

$$\text{Max} \quad f(\boldsymbol{\Delta u}) \tag{5.10}$$

Subject to:

$$\boldsymbol{A}(\boldsymbol{u} + \boldsymbol{\Delta u}) \leq \boldsymbol{b} \tag{5.11}$$

$$\boldsymbol{\Delta u} \geq \boldsymbol{0} \tag{5.12}$$

Given the current lane configuration $\boldsymbol{u}$, we want to expand it as much as possible. $\boldsymbol{\Delta u}$ are the number of increased lanes. $f(\boldsymbol{\Delta u})$ is a function to evaluate total expansion on the network. A straight forward setting is the summation of expansion on all links. Setting f

as the sum of squares would encourage expanding multiple lanes on one link while setting it as the sum of square root would encourage expanding one lane for more links. The lane configuration after the expansion is $\boldsymbol{u} + \boldsymbol{\Delta u}$. Its feasibility is guaranteed by equation 5.11.

The following procedures are pretty standard. We use the total vehicle-hours as the performance measure. After scoring the current population with simulation, crossover and mutation are performed on candidates with better performance. The resulting new population will be first treated to ensure feasibility and encourage the full utilization of available lanes. This customized genetic algorithm process is then repeated until the network stoped improving or the maximum number of iterations has been reached. In order to have a better performance, parameters such as population size, generations, crossover rate and mutation rate need to be carefully tuned.

This algorithm is a direct application of the genetic algorithm. Given the complexity of the problem, it can only work on a toy network with a small feasible region. Moreover, the algorithm does not take advantage of additional information that can be obtained from the simulation that scores the fitness of the candidates. For example, link densities from the simulation is essentially a proxy for "shadow price". Expanding the link with the highest density could potentially improve network performance. Additionally, the customized treatment of utilizing the number of available lanes as much as possible after each iteration could be harmful to the final objective. Some of the increments are useless because of low traffic volume associated with the movement; and increasing the number

of lanes assigned to such movements at an early stage could potentially prevent assigning the lanes to more congested directions or turning movements. Given the discussion above, the customized genetic algorithm would only be used later in the case studies as a benchmark.

5.3.2 Congestion Relief Algorithm

When it comes to a real network, the feasible region could be too big for the genetic algorithm to explore. As discussed in 5. 3.1, the information we got from simulation such as the cumulative link flow and link densities are not utilized at all by the genetic algorithm. Improving the selection of ridesharing location and route recommendations are hard. Because they are highly entangled with each other and depend on lane configurations. To elaborate, if lane configurations are fixed and we are trying to reduce congestion by adjusting route recommendations, it is still hard to quantify the difference in congestion by changing route recommendation for one OD pair. However, if ridesharing location and route recommendations are fixed, it is relatively easy to estimate the difference in congestion by increasing/decreasing one lane. Thus, the solution space can be naturally divided into two parts, one part consists of ridesharing locations and route recommendations, the other part is lane configurations.

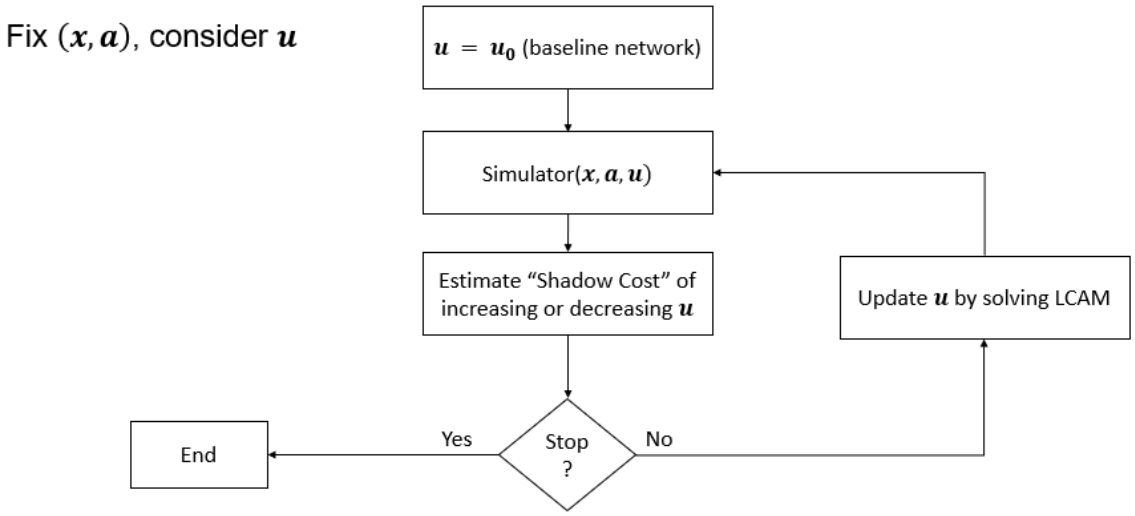

Figure 5-4 Congestion Relief Algorithm

In this section, we propose a congestion relief algorithm iteratively improving the lane configurations for a fixed ridesharing location and route recommendation pair. Congestion relief algorithm has two phases. In the first phase, we propose a method to estimate the congestion increment/decrement on one link by reducing/increasing one lane on its downstream link. The "shadow costs" of changing the lane configuration by one lane for all links are estimated based on cumulative link flow. In the second phase, a lane configuration adjustment model (LCAM) is then formulated to improve current lane configurations based on the estimated "shadow costs". For each fixed pair of route recommendations and ridesharing location, these two phases are conducted iteratively until the adjusted network has already been seen or the maximum number of iterations has been reached. Sections 5.3.2.1 and 5.3.2.2 discuss the two phases in detail.

5.3.2.1 Estimation of "Shadow Costs"

Cumulative link flow by routes is available from the lower level simulation. Based on this information along with the current lane configurations, here we estimate the "shadow costs" of adding or reducing one lane on any link. The calculation is only an approximation of the actual shadow costs because it focuses on the change of the delay (travel time) of any immediate upstream link to the current link, assuming the cumulative arrival flows remain the same. This calculation ignores the network effect of changing the configuration of a link, and is not a comprehensive network sensitivity analysis.

Here we start with the simplest scenario: an intersection with one incoming and one outgoing link. We look at the congestion on the upstream link by changing the number of lanes on the downstream link. Current cumulative arrival and departure on the upstream link are plotted in Figure 5-5. The total vehicle travel time has two parts: free-flow travel time (the area between blue and black lines) and congestion experienced on the upstream link.

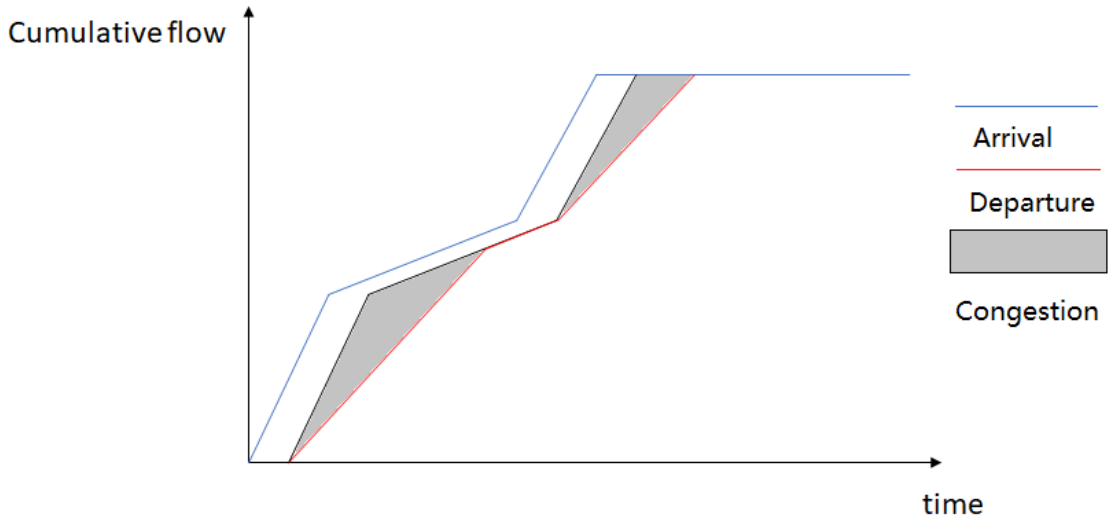

Figure 5-5 Current Cumulative Flows and Congestion

Ignoring other parts of the network, for these two links, increase or decrease the number of lanes (capacity) of downstream would impact the congestion on the upstream link. To illustrate, Figure 5-6 Increase or Decrease One Lane on the Downstream Link shows the cumulative departure after increasing/decreasing one lane on the downstream link.

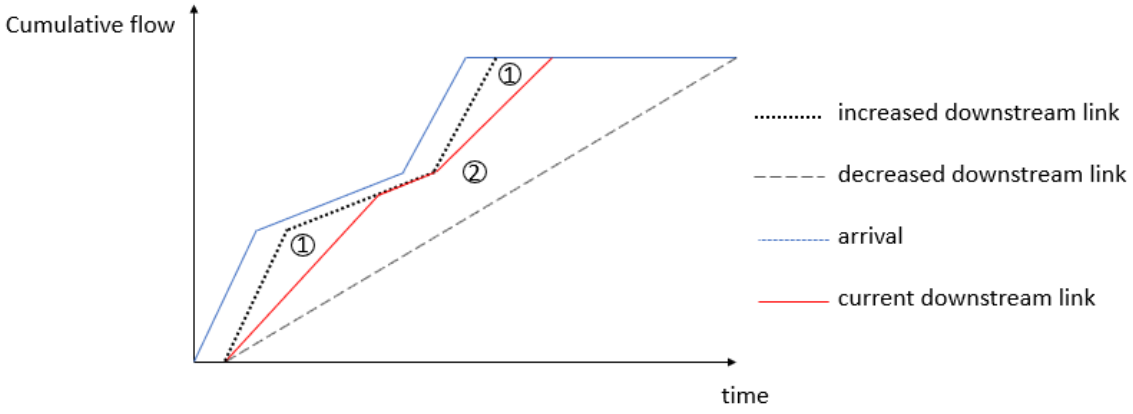

Figure 5-6 Increase or Decrease One Lane on the Downstream Link

Areas labeled by 1 are the congestion reduction (in vehicle-hours) after increasing one lane and area 2 is the congestion caused by decreasing one lane. Thus, the "shadow cost" of increasing one lane of the downstream link is area 1 and the "shadow cost" of decreasing one lane on the downstream link is area 2.

Following the same idea, this estimation can be generated to an intersection with multiple incoming links and multiple outgoing links. And after a similar analysis, we can get increasing prices and decreasing prices for all the outgoing links. The estimation is more complicated because with multiple incoming links, the receiving capacity on the outgoing link should be proportional to the number of incoming lanes. And with multiple outgoing links, once one outgoing link is blocked, then all the incoming links trying to send flow to the blocked link would be blocked. Our network formulation eliminates the many to many intersections which simplify the estimation. But it is still a complicated procedure. The estimation of many-to-one (merging) and one-to-many (diverging) intersections are described in Appendix C.

5.3.2.2 Lane Configuration Adjustment Model

After the estimation of the "shadow cost" for each link, the next step is to adjust the lane configurations to reduce congestion. This is done by solving an integer programming problem. It is worth noting that the "shadow cost" of increasing a lane is always non-positive, indicating congestion reduction, while the "shadow cost" of decreasing a lane is non-negative, indicating increased congestion. Suppose $\boldsymbol{u}$ is the current lane

configuration and $\boldsymbol{Au} \leq \boldsymbol{b}$ are constraints in the optimization model regarding lane configurations. The "shadow costs" for increasing and decreasing a lane are denoted by row vectors $\boldsymbol{in}$ and $\boldsymbol{de}$ respectively. $\boldsymbol{y}$ and $\boldsymbol{z}$ are binary variables indicating whether to increase/decrease one lane on that link. Then the problem can be formulated as follows.

Lane Configuration Adjustment (LCA) Formulation:

$$\text{Min} \quad \boldsymbol{in} * \boldsymbol{y} + \boldsymbol{de} * \boldsymbol{z} \tag{5.13}$$

Subject to:

$$\boldsymbol{A}(\boldsymbol{u} + \boldsymbol{y} - \boldsymbol{z}) \leq \boldsymbol{b} \tag{5.14}$$

$$\boldsymbol{y} + \boldsymbol{z} \leq \mathbf{1} \tag{5.15}$$

$$\Sigma(\boldsymbol{y} + \boldsymbol{z}) \leq K \tag{5.16}$$

The objective function of this optimization model is to minimize the total "shadow cost" by adjusting current lane configurations at most by one lane on each link. Constraint (5.14) guarantees the feasibility after adjusting the lanes. Equation (5.15) states that one link cannot be increased and decreased at the same time. Finally, constraint (5.16) put an upper bound K to the total number of links changed (increased or decreased). The number K can start with a large number; and the problem can be solved multiple times

with reduced K value each time, until the total "shadow cost" cannot be decreased any more.

5.3.2.3 Genetic Algorithm with Congestion Relief

After introducing the congestion relief algorithm, lane configurations can be improved given a pair of ridesharing locations and route recommendations. Then, in the Route recommendations and ridesharing locations space, candidates can be explored by, for example, the genetic algorithm.

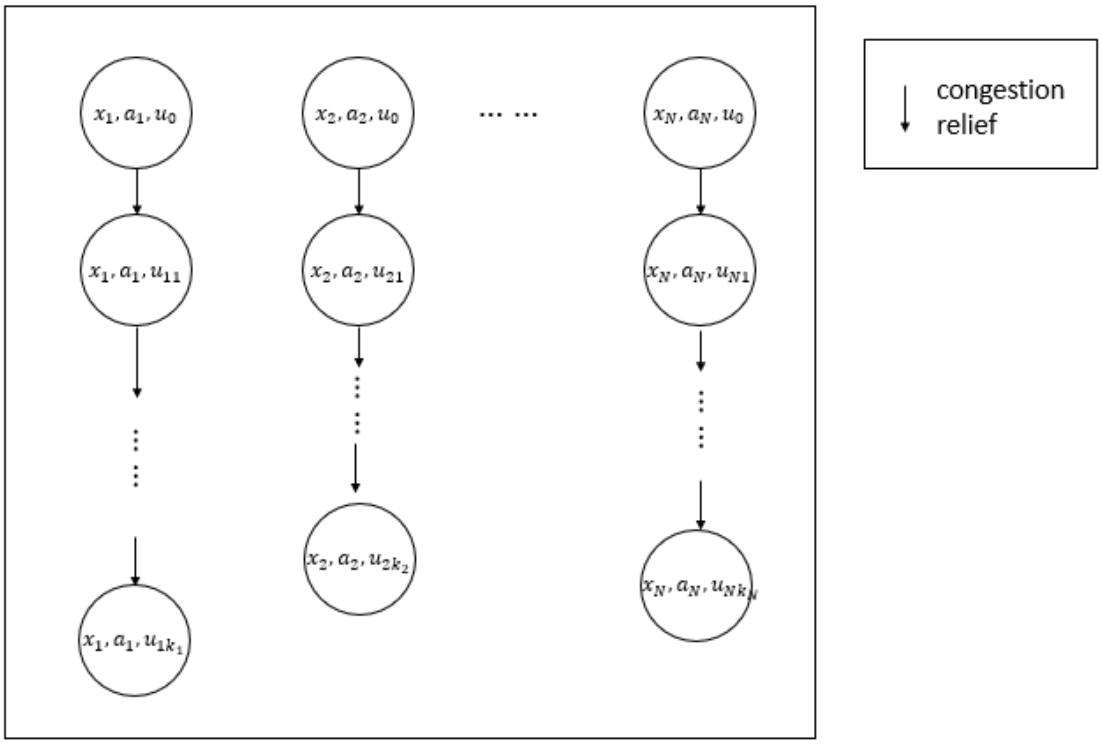

Figure 5-7 One Iteration in Genetic Algorithm with Congestion Relief

Figure 5-7 One Iteration in Genetic Algorithm with Congestion Relief shows one iteration in the genetic algorithm with Congestion Relief. At the beginning of the algorithm, N candidates of the ridesharing locations and route recommendations are generated using genetic algorithm.: $(x_1, a_1), (x_2, a_2), \dots, (x_N, a_N)$. For each pair of ridesharing locations x_i and route recommendations a_i, starting from the default network settings u_0, we estimate the congestion costs and solve the lane configuration adjustment model. Then adjust the network and repeat this process until the maximum number of iterations is met or the adjusted network has already been seen.

5.4 Case Studies

5.4.1 A Toy Network

Our first case study is a small network with simple settings. The true optimal solution can be found through enumeration. The customized genetic algorithm with link expansion described in section 5.3.1 is performed on the small network, and the resulting solution is compared to the true optimum. The case study shows that the GALE generates reasonable travel plans and can greatly reduce total travel time experienced by changing the network structure, route recommendation and drop-off location. The resulting plan's total vehicle-hours is close to the underlying optimal plan for this small network.

The transportation network considered in this case study is a circular network as shown on the left side of Figure 5-7. All the links shown in the physical network graph are bi-directional. Travelers enter or leave this network only through node 1. The special event

takes place at node 12 with a reserved parking facility there. A parking space is guaranteed at node 12 but only certain groups of travelers have access (such as season ticket holders). Two public parking facilities are positioned at nodes 7 and 11 with fixed probabilities of finding a parking space. The free flow speed is set as 40 miles/hour on all links. The simulation duration for each step is 0.0025 hour (9 seconds) which is smaller than the minimum time required for link traversal. The total simulation time is half an hour allowing all the vehicles to finish their trips.

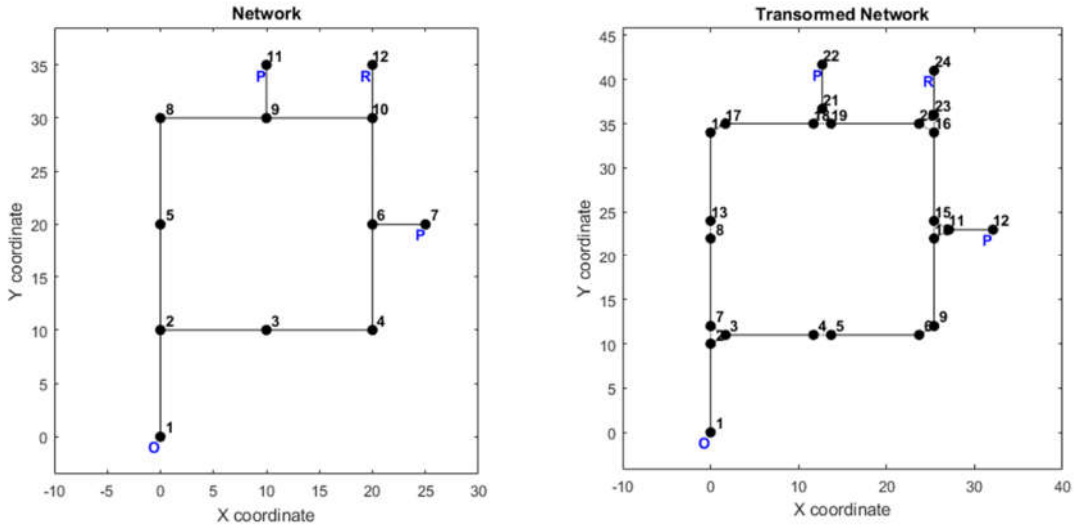

Figure 5-8 Physical Network and Transformed Network

The first step is to establish a network model for the physical network according to methods described in section 5.2.1.1. As shown on the right side of Figure 5-7, after transformation and re-indexing, the special event network model now has 12 nodes and 56 links. Three main blocks connecting parking facilities have a length of 0.5 miles and

other links in the main blocks are 1 mile long. Dashed lines represent transition links with a length of 0.2 miles, modeling the turning areas. The maximum number of lanes on each link is 4 for two directions together. Each transition area also has 4 lanes in total. The flow capacity and jam densities are set as 1200 vehicles per hour per lane and 200 vehicles per mile per lane respectively. The OD pairs considered here are from node 1 to the three parking facilities (node 12, 22, 24) and from public parking facilities (node 12, 22) back to node 1. Node 24 to node 1 is not considered as an OD pair because parking space is guaranteed for the reserved parking facility. The majority of the total demand is from origin to the reserved parking facility. Minor demands are going to public parking nodes and taking ridesharing service. Each OD pair has two alternative routes: clockwise and counterclockwise. The ridesharing path is from node 1 taking a counterclockwise circular route and back to node 1. The probability of finding parking at all public parking facilities are fixed as 0.8.

It only takes about 20 seconds to complete a single run of traffic simulation for this network. Thus, we are able to find the true optimal solution through enumeration. Under the settings described above, the true optimal solution leads to a total objective value of 68.92 vehicle-hours. Since the solution obtained from the genetic algorithm is intrinsically random, 10 experiments are performed with different initial populations, each solved with 5 generations and population size of 20. The table below shows the total vehicle-hours of the best solution in each generation for all 10 experiments:

Table 5-2 Toy Network GALE Results

experiment \ generation	1	2	3	4	5
1	81.83	81.83	70.32	70.32	70.32
2	82.04	82.04	79.18	79.18	79.18
3	95.06	94.92	75.07	75.07	75.07
4	88.09	88.09	87.95	87.95	87.95
5	94.03	75.07	75.07	75.07	75.07
6	95.06	77.46	77.46	77.46	70.46
7	95.06	79.18	79.18	79.18	79.18
8	100.79	77.46	77.46	77.46	77.46
9	88.0	79.18	79.18	70.32	70.32
10	88.09	88.09	88.09	88.09	88.09

From the table above, the best candidates from the initial population of the 10 experiments average a total vehicle-hours of 90.8, which is 32% higher than the true optimal solution. After 5 generations, the best candidate reaches 77.31 vehicle-hours on average, which is only 8% higher than the true optimal solution. Another observation is that, improvement tends to happen in earlier generations. The genetic algorithm could be stuck in a local optimal after a few generations. This can be mediated by increasing the population size and keeping more diversity in each generation.

5.4.2 Super Bowl XLIX

The Super Bowl XILX was held on February 1, 2015 at the University of Phoenix Stadium in Glendale, Arizona. At the time of this writing, it is the most-watched Super Bowl and most watched television program of all time with 114.4 million American viewers. The University of Phoenix Stadium has the capacity to accommodate more than 60000 spectators and has more than 14000 parking spaces. It is a great real-world case study to test the proposed algorithms. Figure 5-9 shows the transportation network surrounding the stadium and the network model after transformation. The network model has 98 nodes and 308 links, among which 98 are main block links and 210 are transition links. The default number of lanes on each link is obtained from OpenStreetMap data API while the missing data are manually made up for by counting manually from the OpenStreetMap satellite photos. The network has 6 origins and 5 destinations (parking lots). We are assuming the route assignment rule $(1-\alpha)$-soft assignment, where α is the compliance rate.

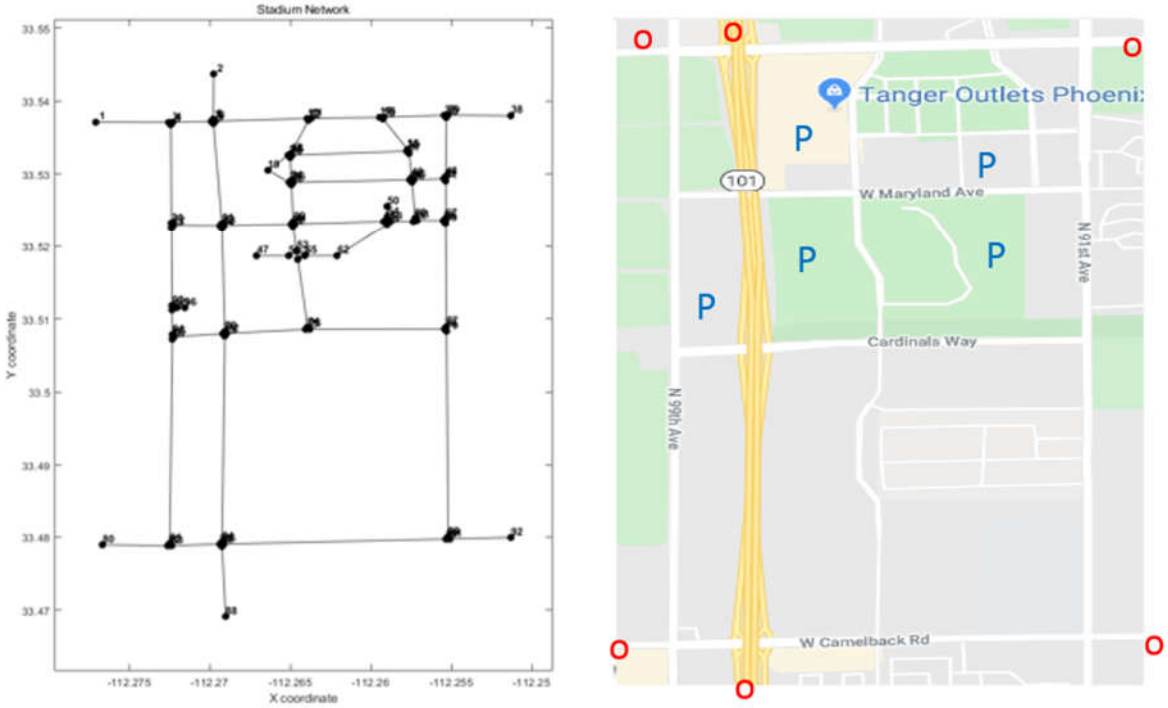

Figure 5-9 Network Model

Before running the algorithm, an important input is still missing – the OD demands. The following three subsections first estimate OD demand based on link flow data during the ingress of Super Bowl XLIX. Starting with a strong assumption where $\alpha = 1$ in section 5.4.2.2, the proposed algorithms are tested on the network. Sensitivity analysis regarding the compliance rate α is conducted in section 5.4.2.3.

5.4.2.1 OD Estimation

Link flow data for some of the links are available during ingress (Table 5-3) from the traffic management center of the City of Glendale, AZ. For OD estimation, we assume

travelers have full information and want to minimize their own travel time. Thus, resulting flow pattern follows user equilibrium (UE) conditions.

Table 5-3 Part of Observed Link Flow

Road Segment	Direction	11-12pm	12-1pm	1-2pm
Glendale Ave. between.	EB	1279	980	1159
Loop 101 & 95th Ave.	WB	1346	1095	945
Maryland Ave. between.	EB	485	473	417
Loop 101 & 95th Ave.	WB	233	159	149
Camelback Rd. between.	EB	1181	1489	1558
95th Ave. & 91st Ave.	WB	1048	1063	1127
99th Ave. between. Glendale	NB	602	715	652
Ave. & Maryland Ave.	SB	633	809	731
99th Ave. between. Bethany	NB	873	1066	969
Home Rd. & Camelback Rd.	SB	619	805	656

For any OD demand $\boldsymbol{d}$, the corresponding UE link flows are calculated by the Frank-Wolf algorithm. The resulting UE link flow is then compared to the observed link flow. Starting from zero demand, we conduct a compass search in the OD demand space trying to minimize the least square error between UE flow and observed hourly average link flow during ingress. In each iteration, a set of candidates is created by adding/reducing the demand of one OD pair by the current step size. We check the least square errors (LSE) of all candidates. If none of the LSE is better than the current LSE, the step size will be reduced before repeating the process; otherwise, the best candidate is selected and process is repeated. The algorithm terminates if the LSE or step size is small enough.

With 6 origins and 5 destinations, 30 OD demands are estimated. The resulting UE flow from the estimated OD demands can approximately replicate the observed link flow and lead to reasonable congestion.

5.4.2.2 Numerical Results

The purpose of this case study is to test the performance of the proposed algorithms on a real-world network. The network has 85 intersections and 98 main block links where the lane configurations can be changed. For each OD pair, 3 shortest paths under free flow condition are calculated as alternative routes. 6 links close to the destination are selected as potential ridesharing link. Both genetic algorithm with link expansion (GALE) and genetic algorithm with congestion relief algorithm (CR) are performed.

For this case study, the probabilities of finding parking at the 5 parking lots are set to 1. The probability values can be easily modified, but we do not have any relevant real data to estimate reasonable probability values. Lane-changing intensity ϵ for ridesharing links is set as 1.1. The benchmark settings of route recommendations and ridesharing location is recommending the shortest path for each OD pair and select the nearest link as ridesharing location. The interval for each timestep in the simulation is set at 3.6 seconds. This is a relatively small timestep and would lead to longer computational time to run the simulation. However, since the purpose of this case study is to verify the performance of the proposed algorithms on real network, we use this smaller timestep for more realistic outcome from the simulation. For faster computation time, a longer timestep can be used in the future. The simulation duration is set as 30 minutes mimicking the most intensive

arrivals right before the event and the total number of vehicles entered the network is about 1400. Ridesharing vehicles from different origins will take the shortest path to ridesharing link then leave the network by the shortest path to the nearest edge node. The population of GALE is set to 100. One simulation takes about 20 minutes. With 10 processors running in parallel, one iteration for GALE takes about 200 minutes. The running time for GACR depends on the number of iterations of GA and CR algorithms. We set the maximum number of iterations of CR as 5. The maximum number of iterations is network-specific and should be large enough for most route recommendation-ridesharing location pairs to evolve to its optimal lane configurations. Intuitively, it should be greater than the maximum number of possible bottlenecks on the routes. The population size of the genetic algorithm with GACR is set as 20 to make the running times for GACR and GALE are comparable. Since the solutions to be explored by GACR is way smaller than GALE, taking 20 candidates from GACR's solution space is already covering more than taking 100 candidates from GALE's solution space.

Table 5-4 GALE and GACR Performance (in Vehicle-hours)

Iteration	1	2	3	4	5
GALE	167.36	167.36	166.46	166.46	166.46
GACR	110.10	101.62	101.62	101.43	101.08

GALE is not able to improve the objective function significantly, due to the large feasible region. After 5 iterations, the total vehicle-hours is improved by 0.9. And the best

performance is achieved based on the benchmark network which gives 167.36 vehicle-hours and later improved to 166.46 vehicle-hours. The populations from one iteration to another are already converging, so keep running the algorithm would not likely to further improve the result. After a closer look into the population pool from each iteration, the best plans in each iteration are all generated by mutation or crossover of the default lane setting. Other lane configurations in the initial population failed to provide competing performance. This is because: 1) the solution space is still too large. With 6 options for ridesharing location, 30 OD pairs and 308 links, the total number of feasible solutions is easily over $6 \times 3^{30} \times 2^{308}$. This makes it impossible to explore the whole space. 2) The built-in complexity of the problem. Route recommendations and lane configurations are tangling with each other. A good pair of ridesharing location and route recommendation (for example, recommended routes have fewer conflicts with each other) depends highly on the lane configurations. For the same ridesharing location and route recommendations, its fitness cannot be properly estimated without the right lane configurations.

On the other hand, GACR is able to generate a good solution after the first iteration, saving 34% of total vehicle-hours compared to the baseline network (167.36 vehicle-hours). This value keeps decreasing along with the outer iteration (total vehicle-hours is improved to 100.54 after two more iterations). CR did a good job optimizing the network with given route recommendations and ridesharing location. However, the optimal plan depends highly on the initial population. For real world applications, it would be better to have experienced planners propose some potential ridesharing options and routes.

Applying GACR to these plans can then help optimize the networks and provide estimates of total travel times for each plan. This also leaves the question of how to initialize the population in a more strategic way as a future research direction.

It is also observed that other than the optimal plan, GACR is also providing some competitive plans with performance close enough to the optimal value. We would link to see how the performances will change if this assumption is relaxed to be more realistic. These plans, together with the optimal plan, are tested in Section 5.4.2.3 against more realistic route assignment heuristics for robustness.

5.4.2.3 Sensitivity Analysis

In the previous section, GACR is able to generate multiple plans with similar performances under the assumption that all the travelers follow the recommended routes. However, in the real world, there are always travelers who would take alternative routes. In this section, taking 5 plans generated by GACR that has close performance compared to the best plan, we test their robustness against various traffic assignment rules; that is, how their performance will change if only some of the travelers followed the recommended routes. We assume a compliance rate α that represents the portion of travelers following recommended routes. (α was 1 in section 5.4.2.2) And the rest of the travelers choose alternative routes (3 shortest paths other than the recommended route) with equal probabilities.

Table 5-5 Sensitivity Analysis Regarding Compliance Rate

	$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.33$
Plan 1	100.54	100.45	102.35	111.84	117.20	127.38
Plan 2	100.73	100.62	102.48	111.90	117.20	127.38
Plan 3	102.07	101.56	103.13	111.09	116.61	127.63

The generated PSE traffic plans are able to tolerate some uncertainty in the compliance rate. Their performances are even slightly better when $\alpha = 0.9$ compared to $\alpha = 1$. That's because the alternative routes that are not recommended can still handle minor demands even though the lane configurations are not designed for these routes.

5.5 Summary

In this chapter, we propose a bi-level model combing optimization and simulation approaches. The upper level is an optimization model aiming to minimize total travel time experienced by travelers. The decision variables are the number of lanes on each link, route recommendations, and ridesharing drop-off locations. In the lower level, an LTM incorporating parking and ridesharing is used to evaluate the total travel time and provide feedback to the upper level. The proposed methodology explicitly modeling the traffic dynamics of parking searching and ridesharing on a network during a special event. An efficient congestion relief algorithm is proposed to solve this model. The algorithm iteratively improves lane configurations give fixed ridesharing locations and route

recommendations. The model and algorithm are further tested on a real-world network centered at the University of Phoenix Stadium during the Super Bowl. The resulting optimal plan is able to save 34% of the total vehicle-hours compared to default lane configurations. Sensitivity analysis has also been conducted regarding the compliance rate of travelers following recommended routes.

6 CONCLUSION AND FUTURE RESEARCH

6.1 Conclusion

To conclude, this dissertation investigates congestion mitigation during the ingress of a planned special event (PSE). Starting from the fundamental information, a parking occupancy prediction framework is proposed to forecast future parking occupancy from historical occupancy data alone. The framework is a model-based approach grounded in the underlying stochastic queuing process and validated by real data. Furthermore, we investigated introducing learning algorithms into parking guidance and information systems that employ a central server, in order to provide estimated user-optimal parking searching strategies to individual travelers. The central server collects processes and distributes data regarding parking search process. It runs a modified Q-learning approach taking advantage of the network topology and dramatically reduces the size of the problem. Our numerical experiments have demonstrated that the proposed algorithm is able to produce high quality solutions with limited training data, and thus has great potential to be applied in real time. Finally, we propose a bi-level model combing optimization and simulation approaches. The upper level is an optimization model aiming to minimize total travel time experienced by travelers. The decision variables are the number of lanes on each link, route recommendations, and ridesharing drop-off locations. In the lower level, a link transmission model (LTM) incorporating parking and ridesharing is used to evaluate the total travel time and provide feedback to the upper level. We also developed effective and efficient heuristic solution algorithms for the PSE

traffic planning problem. A case study of Super Bowl XLIX held in Glendale, AZ in 2015 was conducted. The results show that our methods and approaches are able to produce an effective comprehensive traffic plan with reasonable computation time.

6.2 Future Research

While chapter 3 explored methods for parking occupancy prediction when a facility is congested due to abnormal parking demand (such as a special event), the dataset we currently have does not seem to be from facilities that are recurrently over-saturated. We plan to collect more data from additional facilities such as on-street parking facilities that are more likely to be recurrently over-saturated. If we are successful in obtaining such data, our future research will investigate how appropriate the proposed direct estimation method (as described in Section 3.3.2.1) is for such facilities. Additionally, the model-based methods in our framework allow us to tie in the parking availability estimation with network traffic and driver behaviors through the arrival and departure rates. This study is the first step towards being able to incorporate the parking searching process (reflected in the stochastic arrival process) in parking occupancy prediction, as well as to investigate network traffic effects as a result of parking availability information provision and the subsequent possibly altered parking search behaviors using game-theoretical approaches.

There are several interesting future research directions in chapter 4. The determination of reasonable time windows is not addressed in chapter 4. We briefly discussed possible approaches in Section 4.1. Further development and analysis of these approaches would

contribute to more comprehensive parking guidance and information service framework towards implementation. It would also be interesting to see how the different approaches to handling recommendation updates discussed in Section 4.1 would affect the performance of the system. Improvements to the learning algorithm itself might also be a possibility. In this chapter, the centralized server is considered to have no *a priori* knowledge of the probabilities of finding parking. In some circumstances, we may be able to retrieve some *a priori* knowledge from available information even before training starts. This could further reduce the amount of data required and the proposed algorithm will converge faster to the user-optimal strategy. Another possibility is to relax the fixed probability within a time window into a time-dependent probability. No matter how fast a server could learn, the parking conditions may change constantly. How to generate time-dependent searching strategies from learning needs further investigation. Finally, the authors plan to incorporate other components into the parking network such as parking reservations, pricing and ridesharing and analyze their impact on the transportation network.

A future direction to explore is the exploration in ridesharing location and route recommendation space. Currently this is done by a straightforward application of the genetic algorithm. However, the genetic algorithm is not capable to understand the interactions between routes. Finding a systematic and effective method to generate reasonable candidates in ridesharing location and route recommendation space would greatly reduce the number of iterations needed. Also, more realistic considerations should

be considered such as vehicle-pedestrian conflicts and right-turn preference. Pedestrians have a huge impact on special event networks. Pedestrians crossing streets would not only increase congestion but also cause potential safety issues. Thus, the pedestrian routes from drop-off locations to final destination should cross as fewer roads as possible. In the current settings, left and right turns are treated equally in this dissertation. But in reality, turning left through oncoming traffic could take much longer than right turns.

REFERENCES

Abdulhai, Baher, Rob Pringle, and Grigoris J. Karakoulas. 2003. "Reinforcement Learning for True Adaptive Traffic Signal Control." Journal of Transportation Engineering 129(3): 278–85.

Alexander, Lauren P, and Marta C González. 2015. "Assessing the Impact of Real-Time Ridesharing on Urban Traffic Using Mobile Phone Data." Proceedings ofUrbComp'15: 1–9.

Arel, I., C. Liu, T. Urbanik, and A.G. Kohls. 2010. "Reinforcement Learning-Based Multi-Agent System for Network Traffic Signal Control." IET Intelligent Transport Systems 4(2): 128. http://digital-library.theiet.org/content/journals/10.1049/iet-its.2009.0070.

Arentze, T. A., and H.J.P. Timmermans. 2003. "Modeling Learning and Adaptation Processes in Activity- Travel Choice: A Framework and Numerical Experiments." Transportation 30(1): 37–62.

Asakura, Y., and M. Kashiwadani. 1994. "Effects of Parking Availability Information on System Performance: A Simulation Model Approach." Proceedings of VNIS'94 - 1994 Vehicle Navigation and Information Systems Conference: 251–54. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=396832.

Benenson, Itzhak, Karel Martens, and Slava Birfir. 2008. "PARKAGENT: An Agent-Based Model of Parking in the City." Computers, Environment and Urban Systems 32(6): 431–39. http://linkinghub.elsevier.com/retrieve/pii/S0198971508000689.

Boyles, Stephen. 2014. "Modeling Parking Search on a Network Using." Transportation Research Board 93rd Annual Meeting. January 12-16, Washington, D.C.

Cai, Chen, Chi Kwong Wong, and Benjamin G. Heydecker. 2009. "Adaptive Traffic Signal Control Using Approximate Dynamic Programming." Transportation Research Part C: Emerging Technologies 17(5): 456–74. http://dx.doi.org/10.1016/j.trc.2009.04.005.

Caicedo, Felix, Carola Blazquez, and Pablo Miranda. 2012. "Prediction of Parking Space Availability in Real Time." Expert Systems with Applications 39(8): 7281–90. http://dx.doi.org/10.1016/j.eswa.2012.01.091.

Cao, Jin, and Monica Menendez. 2015. "System Dynamics of Urban Traffic Based on Its Parking-Related-States." Transportation Research Part B: Methodological 81(August): 718–36. http://dx.doi.org/10.1016/j.trb.2015.07.018.

Chan, Nelson D, and Susan A Shaheen. 2012. “Ridesharing in North America : Past , Present , and Future.” 1647(1): 93–112.

Chaniotakis, Emmanouil, and Adam J Pel. 2015. “Drivers ’ Parking Location Choice under Uncertain Parking Availability and Search Times : A Stated Preference Experiment.” Transportation Research Part A 82: 228–39. http://dx.doi.org/10.1016/j.tra.2015.10.004.

Dai, Xiaohui et al. 2005. “An Approach to Tune Fuzzy Controllers Based on Reinforcement Learning for Autonomous Vehicle Control.” IEEE Transactions on Intelligent Transportation Systems 6(3): 285–93.

Desjardins, C, and B Chaib-draa. 2011. “Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach.” IEEE Transactions on Intelligent Transportation Systems 12(4): 1248–60.

Dolan, Elizabeth D., and Jorge J. Moré. 2002. “Benchmarking Optimization Software with Performance Profiles.” Mathematical Programming, Series B 91(2): 201–13.

Federgruen, A., P.J. Schweitzer, and H.C. Tijms. 1978. “Contraction Mappings Underlying Undiscounted Markov Decision Problems.” Journal of Mathematical Analysis and Applications 65: 711–30.

Firebase. “SF Parking Open Data Sets - Firebase.”

Furuhata, Masabumi et al. 2013. “Ridesharing: The State-of-the-Art and Future Directions.” Transportation Research Part B: Methodological 57: 28–46. http://dx.doi.org/10.1016/j.trb.2013.08.012.

Geng, Yanfeng, and Christos G. Cassandras. 2012. “A New ‘Smart Parking’ System Infrastructure and Implementation.” Procedia - Social and Behavioral Sciences 54(October 2012): 1278–87. http://linkinghub.elsevier.com/retrieve/pii/S1877042812043042.

Geroliminis, Nikolas. 2015. “Cruising-for-Parking in Congested Cities with an MFD Representation.” Economics of Transportation 4(3): 156–65. http://dx.doi.org/10.1016/j.ecotra.2015.04.001.

Giuffrè, Tullio, Sabato Marco Siniscalchi, and Giovanni Tesoriere. 2012. “A Novel Architecture of Parking Management for Smart Cities.” Procedia - Social and Behavioral Sciences 53: 16–28. http://linkinghub.elsevier.com/retrieve/pii/S1877042812043182.

Goerigk, Marc, Kaouthar Deghdak, and Philipp Hebler. 2014. “A Comprehensive Evacuation Planning Model and Genetic Solution Algorithm.” Transportation Research Part E: Logistics and Transportation Review 71: 82–97.

Ibeas, A., L. dell’Olio, M. Bordagaray, and J. de D. Ortúzar. 2014. “Modelling Parking Choices Considering User Heterogeneity.” Transportation Research Part A: Policy and Practice 70: 41–49. http://www.sciencedirect.com/science/article/pii/S0965856414002341.

IBM. 2011. “Global Parking Survey: Drivers Share Worldwide Parking Woes.”

Idris, M.Y.I. et al. 2009. “Car Park System: A Review of Smart Parking System and Its Technology.” Information Technology Journal 8: 101–13.

Jin, Wen-long. 2010. “A Kinematic Wave Theory of Lane-Changing Traffic Flow.” Transportation Research Part B: Methodological 44(8–9): 1001–21. http://dx.doi.org/10.1016/j.trb.2009.12.014.

Kim, Sangho, Shashi Shekhar, and Manki Min. 2008. “Contraflow Transportation Network Reconfiguration for Evacuation Route Planning.” IEEE Transactions on Knowledge and Data Engineering 20(8): 1115–29.

Kwon, Jaimyoung, Michael Mauch, and Pravin Varaiya. 2007. “Components of Congestion: Delay from Incidents, Special Events, Lane Closures, Weather, Potential Ramp Metering Gain, and Excess Demand.” Transportation Research Record: Journal of the Transportation Research Board 1959(1959): 84–91.

Latoski, S. P. et al. 2003. Managing Travel for Planned Special Events.

Leclercq, Ludovic, Alméria Sénécat, and Guilhem Mariotte. 2017. “Dynamic Macroscopic Simulation of On-Street Parking Search: A Trip-Based Approach.” Transportation Research Part B: Methodological 101: 268–82.

Levy, Nadav, Marc Render, and Itzhak Benenson. 2015. “Spatially Explicit Modeling of Parking Search as a Tool for Urban Parking Facilities and Policy Assessment.” Transport Policy 39: 9–20. http://dx.doi.org/10.1016/j.tranpol.2015.01.004.

Li, Ziru, Yili Hong, and Zhongju Zhang. 2016. “Do Ride-Sharing Services Affect Traffic Congestion? An Empirical Study of Uber Entry.” SSRN Electronic Journal (2002): 1–29.

Melo, Francisco S. 2001. “Convergence of Q-Learning: A Simple Proof.” Institute Of Systems and Robotics, Tech. Rep: 1–4.

Meng, Qiang, Hooi Ling Khoo, and Ruey Long Cheu. 2007. "Microscopic Traffic Simulation Model-Based Optimization Approach for the Contraflow Lane Configuration Problem." Journal of Transportation Engineering 134(1): 41–49.

Millard-Ball, Adam, Rachel R. Weinberger, and Robert C. Hampshire. 2014. "Is the Curb 80% Full or 20% Empty? Assessing the Impacts of San Francisco's Parking Pricing Experiment." Transportation Research Part A: Policy and Practice 63: 76–92.

Van Ommeren, Jos N., Derk Wentink, and Piet Rietveld. 2012. "Empirical Evidence on Cruising for Parking." Transportation Research Part A: Policy and Practice 46(1): 123–30. http://dx.doi.org/10.1016/j.tra.2011.09.011.

Prashanth L. A., and Shalabh Bhatnagar. 2011. "Reinforcement Learning With Function Approximation for Traffic Signal Control." Intelligent Transportation Systems, IEEE Transactions on 12(2): 412–21.

Rajabioun, Tooraj, Brandon Foster, and Petros Ioannou. 2013. "Intelligent Parking Assist." 2013 21st Mediterranean Conference on Control and Automation, MED 2013 - Conference Proceedings: 1156–61.

Rajabioun, Tooraj, and Petros A Ioannou. 2015. "On-Street and Off-Street Parking Availability Prediction Using Multivariate Spatiotemporal Models." IEEE Transactions on Intelligent Transportation Systems 16(5): 2913–24.

Rayle, Lisa et al. 2016. "Just a Better Taxi? A Survey-Based Comparison of Taxis, Transit, and Ridesourcing Services in San Francisco." Transport Policy 45: 168–78. http://dx.doi.org/10.1016/j.tranpol.2015.10.004.

Ruan, Jin-mei et al. 2016. "How Many and Where to Locate Parking Lots ? A Space – Time Accessibility-Maximization Modeling Framework for Special Event Traffic Management." Urban Rail Transit 2(2): 59–70.

Schwartz, Anton. 1993. "A Reinforcement Learning Method for Maximizing Undiscounted Rewards." Machine Learning Proceedings 1993 298: 298–305.

Šemrov, D. et al. 2016. "Reinforcement Learning Approach for Train Rescheduling on a Single-Track Railway." Transportation Research Part B: Methodological 86: 250–67.

Shoup, Donald C. 2006. "Cruising for Parking." Transport Policy 13(6): 479–86.

Silvers, JR. 2012. Professional Event Coordination. John Wiley & Sons.

SUTTON, RICHARD S.. BARTO. 1998. REINFORCEMENT LEARNING : An Introduction. Cambridge: MIT press. https://mitpress.mit.edu/books/reinforcement-learning-second-edition.

Tuydes, Hediye, and Athanasios Ziliaskopoulos. 2007. "Tabu-Based Heuristic Approach for Optimization of Network Evacuation Contraflow." Transportation Research Record: Journal of the Transportation Research Board 1964: 157–68.

Vickrey. 1994. "1994_Vickrey_Statement to the Joint Committee on Washington DC Problems_J Urban Economics.Pdf." Journal of urban economics 36(1): 42–65.

Watkins, Christopher John Cornish Hellaby. 1989. Doctoral dissertation, King's College, Cambridge "Learning from Delayed Rewards."

Wolshon, Brian, and Laurence Lambert. 2004. Convertible Roadways and Lanes: A Synthesis of Highway Practice.

Wong, C. K., and S. C. Wong. 2003. "Lane-Based Optimization of Signal Timings for Isolated Junctions." Transportation Research Part B: Methodological 37(1): 63–84.

Xiao, Jun, Yingyan Lou, and Joshua Frisby. 2018. "How Likely Am I to Find Parking? – A Practical Model-Based Framework for Predicting Parking Availability." Transportation Research Part B: Methodological 112: 19–39. https://www.sciencedirect.com/science/article/pii/S0191261517301686.

Xie, Chi, Dung Ying Lin, and S. Travis Waller. 2010. "A Dynamic Evacuation Network Optimization Problem with Lane Reversal and Crossing Elimination Strategies." Transportation Research Part E: Logistics and Transportation Review 46(3): 295–316. http://dx.doi.org/10.1016/j.tre.2009.11.004.

Xie, Chi, and Mark A. Turnquist. 2011. "Lane-Based Evacuation Network Optimization: An Integrated Lagrangian Relaxation and Tabu Search Approach." Transportation Research Part C: Emerging Technologies 19(1): 40–63. http://dx.doi.org/10.1016/j.trc.2010.03.007.

Yperman, Isaak. 2007. Meeting of the Transportation Research Board "Link Transmission Model for Dynamic Network Loading."

APPENDIX A

EXPECTATION AND VARIANCE OF THE OCCUPANCY IN THE CONTINUOUS-TIME MODEL

Let h denote an infinitesimal amount of time, N_t the current occupancy. The following transition probabilities hold (Ross, 2014):

$$P(N_{t+h} = N_t + 1) = \lambda h$$

$$P(N_{t+h} = N_t - 1) = N_t \mu h$$

$$P(N_{t+h} = N_t) = 1 - (\lambda + N_t \mu)h$$

$$P(N_{t+h} = i) = 0, \qquad \forall i \in N, i \neq N_t, N_t - 1, N_t + 1$$

From the equations above, we have:

$$E(N_{t+h}|N_t) = N_t + (\lambda - N_t \mu)h$$

$$\Rightarrow E_{t+h} \coloneqq E(N_{t+h}) = E_t + (\lambda - E_t \mu)h$$

When $h \to 0$, the above equation becomes:

$$E_t' = \lambda - \mu E_t$$

The solution to the partial differential equation above is:

$$E_t = e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu}$$

Similar to the derivation in Appendix B, the variance can be calculated from conditional expectation and variance.

$$V\big(E(N_{t+h}|N_t)\big) = V(N_t + \lambda h - \mu h N_t) = (1 - \mu h)^2 V_t$$

$$E\big(V(N_{t+h}|N_t)\big) = E(\lambda h + \mu h N_t) = \lambda h + \mu h\left(e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu}\right)$$

$$\Rightarrow V_{t+h} \coloneqq V(N_{t+h}) = E\big(V(N_{t+h}|N_t)\big) + V\big(E(N_{t+h}|N_t)\big)$$

$$= (1 - \mu h)^2 V_t + \lambda h + \mu h\left(e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu}\right)$$

$$V_t' = \lim_{h\to 0}\frac{V_{t+h}-V_t}{h} = \lim_{h\to 0}\frac{(1-uh)^2-1}{h}V_t + 2\lambda + \mu e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right)$$

$$= -2\mu V_t + 2\lambda + \mu e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right)$$

$$\Rightarrow V_t = e^{-2\mu t}(V_0 - E_0) + e^{-\mu t}\left(E_0 - \frac{\lambda}{\mu}\right) + \frac{\lambda}{\mu}$$

APPENDIX B

AN ALTERNATIVE DISCRETE-TIME MODEL

At the core of the proposed predictive framework in this manuscript, a queuing model is employed to describe the stochastic occupancy change of a parking facility. The underlying queuing model can be any reasonable model. The main manuscript has demonstrated the proposed framework with the well-established continuous-time Markov M\M\C\C queue. This appendix will further discuss an alternative discrete-time model that focuses on the occupancy at a series of discrete time points with time interval Δt.

B.1 The discrete-time model

Instead of Poisson arrival and departure, both processes can be reasonably modeled as binomial distributions. This could potentially allow us to incorporate the parking-searching process into our framework, which can potentially be more computationally efficient. For arrivals, the probability that an approaching vehicle would select a given parking facility can be treated as a function of traffic and parking facility characteristics (and possibly driver characteristics). This approach, however, would require traffic flow data, which was not readily available for this study. On the other hand, the number of departure from a parking facility during Δt can be calculated from the number of parked vehicles and a leaving probability p. In view of data availability, we will demonstrate a discrete model with Poisson arrival and binomial departure in this appendix.

The number of arrivals during Δt is assumed to be Poisson with average λ. The probability of k vehicles arriving during time interval $[t, t + \Delta t)$, denoted as $P_a(k, \lambda)$ can be written as:

$$P_a(k,\lambda) = P(A(t) = k) = \begin{cases} \frac{\lambda^k}{k!} e^{-\lambda} & if\ k \in \{0\} \cup N \\ 0 & else \end{cases} \quad \text{(B.1)}$$

where $A(t)$ is the number of arrival during $[t, t + \Delta t]$. Note that not all arriving vehicles are guaranteed to find a parking spot when the parking lot becomes full. In this case, we assume these vehicles are blocked and will leave for an alternative parking lot immediately.

Let $D(t)$ denote the number of departures during time interval $[t, t + \Delta t)$; n the occupancy of the parking lot at time t; p the leaving probability of each vehicle in the parking lot during time interval Δt. The probability of k vehicles leaving in Δt, given occupancy n and leaving probability p can then be expressed as:

$$P_d(k,n,p) = P(D(t) = k) = \begin{cases} \binom{n}{k} p^k (1-p)^{n-k} & if\ \ k = 0,1,\dots,n \\ 0 & else \end{cases} \quad \text{(B.2)}$$

It is worth noticing that the average parking time (in terms of Δt) is equal to $1/p$. It is assumed that each vehicle's parking time is greater than Δt. Therefore, Δt should be set to a small enough value. In fact, when occupancy is sufficiently large and the leaving probability sufficiently small, the binomial distribution is a close approximation to Poisson distribution. This is also a well-known conclusion in probability theory.

With the above assumptions, the Markov matrix P of this alternative discrete-time model can be derived. Let $P_{i,j}$ denote the transition probability from occupancy $= i$ to j in time interval $[t, t + \Delta t)$.

$$P_{i,j} = \sum_{k=0}^{i} P_d(k, i, p) \times P_a(k + j - i, \lambda) \qquad i, j = 0,1,2 \dots \dots, C-1 \tag{B.3a}$$

$$P_{i,C} = 1 - \sum_{k=0}^{C-1} P_{i,k} \qquad i = 0,1,2, \dots \dots, C \tag{B. 3b}$$

Note that arrival and departure as events can occur at any time and not necessarily at discrete time points. Rather, the discrete-time model assumes that the arrival and leaving probabilities can only change at discrete time points.

If the parking facility is under-saturated, similar results on time-dependent expectation and variance can be derived for the discrete model:

$$E_m = (1-p)^m \left(E_0 - \frac{\lambda}{p}\right) + \frac{\lambda}{p} \tag{B. 4}$$

$$V_m = (1-p)^{2m}(V_0 - E_0) + (1-p)^m \left(E_0 - \frac{\lambda}{p}\right) + \frac{\lambda}{p} \tag{B. 5}$$

When Equation (3.1) from the continuous-time model is adopted in the regression method, the resulting expectation curve would be the same as that of the equation (B.4) from the discrete-time model, although the estimated arrival and departure rates will not be the same. It is trivial to prove that from the same data set, the discrete-time model will lead to a higher estimated arrival rate and a lower estimated average parking time. But the fact that the expectation curves from the two models are the same means that the occupancy prediction will not be different.

B.2 Computational efficiency of transition probability matrix evaluation

Computational efficiency becomes relevant when it is necessary to evaluate the transition probability matrix. This not only occurs in parameter estimation when a facility is over-saturated, but also in probabilistic prediction of future occupancy regardless of whether a facility is over- or under-saturated. In a smart parking management scenario, the former is likely carried out by a server while the latter on the client's end to reduce communication cost.

In the continuous-time model, calculating the transition probability matrix involves evaluation of a matrix exponential operator $e^X := \sum_{k=0}^{\infty} X^k/k!$ (Coddington, 1955), which is non-trivial. Moler and Loan (2003) provided an extensive review on various approaches to approximating matrix exponential. The applicability of the approaches depends on the matrix properties such as its 2-norm and eigenvalues. Different approaches also vary in reliability, stability, and accuracy. Moler and Loan (2003) identified the scaling-and-squaring method a potential "best" method as it is regarded generally applicable and has several reliable implementations. The computational complexity of the method is about $O\big((m + 1/3)n^3\big)$ floating point operations[3], where m is an integer ranging from 1 to about 15 depending on the 2-norm of the matrix and the desired error bound and n is the size of the square matrix. In our application of modeling

[3] As defined in Moler and Loan (2003), one floating-point computation "involves one floating point multiplication, one floating point addition, a few subscript and index calculations, and a few storage references".

parking process, n is the capacity of the parking facility plus one. Furthermore, note that in the continuous-time Markov model, the arrival and departure rates can also vary with time. In this case, solving for $P(t)$ becomes a linear ordinary differential equation with time-varying coefficients whose closed-form solution does not exist (Blanes, 2009).

For the alternative discrete-time model, the computational complexity of evaluating the transition probability matrix from equations (B.1) – (B.3) is dominated by Equation (B.3). In equations (B.1) and (B.2), the exponentials only need to be calculated once; and the factorials and binomial coefficients can be calculated iteratively. For example, to compute $P_a(k,\lambda)\ \ k = 0,1,2,\dots C$ in formula (B.1), we only need to compute $e^{-\lambda}$ once when calculating $P_a(0,\lambda)$. We then apply $P_a(k+1,\lambda) = \frac{\lambda}{k+1}P_a(k,\lambda)$ to compute all the following probabilities iteratively. The total complexity of equation (3.1) is $O(C)$ plus the computation of $e^{-\lambda}$. Similarly, the complexity of equation (B.2) is also $O(C)$. On the other hand, it is not difficult to see that Equation (B.3) requires $O(C^3/3)$ floating point operations. Therefore, the complexity of evaluating equations (1) – (3) is $O(C^3/3)$, which is lower than the $O\big((m+1/3)C^3\big)$ from the continuous-time model. This shows that the discrete-time model is potentially more computationally efficient than the continuous-time model. The actual computer time needed would depend on the implementation of both approaches and the values of model parameters.

We performed numerical experiments on a DELL Precision T3610 workstation with 2.6GHz CPU and 32GB RAM running Windows 8.1 Pro to compare the actual computation time of the transition probability matrix using both models. We

implemented the discrete-time model in Matlab; for the continuous-time model, we adopted the Matlab built-in function expm() implementing the scaling-and-squaring method with Padé approximation. Our experiment results show that the discrete-time model performs better when the capacity is under 50 while the continuous-time model runs faster when the capacity is higher than 70. When the values of λ and p change, the computation time also changes. If the prediction is carried out by a device at a client's end (i.e., a typical smart phone), one possible approach to boost computational time is to reduce the number of significant digits by aggregating parking occupancy into different bins (for example a bin could consist 10 spots). This way, the size of the transition probability matrix is also reduced by the same factor and the discrete-model can be applied for faster computation for smaller matrices.

APPENDIX C

ESTIMATING SHADOW COST FOR MERGING AND DIVERGING NODES

C.1 Merging Nodes

In this scenario, there are n incoming links and just one outgoing link. Adjusting the number of lanes on the outgoing link is changing the capacity on the link. The simulation period is divided into small time slices with length Δt. During each time slice, if all the incoming links are sending flow with infinite capacity, the flow that is able to go through the intersection would be proportional to number of lanes on incoming links. However, if one incoming link sends less flow than its share. Then its remaining capacity will be shared by other active links. For simplicity, free flow travel time is assumed as 0 since they don't affect the increasing/decreasing cost.

Following the logic above, cumulative departure flow can be calculated on all incoming links after increasing/decreasing the outgoing link. Then, the impact in terms of congestion can be easily calculated.

Pseudocode for cumulative departure on incoming links:

Assuming K incoming links and 1 outgoing links, Increase the number of outgoing lanes by 1 (decreasing is the same), get new outgoing capacity. Calculate maximum receiving flow during Δt by $RF = updated\ capacity * \Delta t$

$C^{up}(k,t)$ is cumulative arrival for link k,

Define $speed_k$ for incoming links as the number of lanes on that link

For t from 1 to $\frac{T}{\Delta t}$:

set $C^{down}(k,t) = C^{down}(k,t-1)$

Calculate sending flow $SF(k) = C^{up}(k,t) - C^{down}(k,t-1)$

Find active incoming links where $SF(k) > 0$

while the number of active sending links > 0:

Update receiving speed $speed_{out}$= summation of active incoming links' speed

Compare $\frac{SF(k)}{speed_k}$ over active links and $\frac{RF}{speed_{out}}$

If $\frac{RF}{speed_{out}}$ is smaller:

update $C^{down}(k,t) = C^{down}(k,t) + \frac{SF(k)}{speed_k} * \frac{RF}{speed_{out}}$,

continue;

else

update $C^{down}(k,t) = C^{down}(k,t) + \min_k \frac{SF(k)}{speed_k} * \frac{SF(k)}{speed_k}$

$SF(k) = SF(k) - \min_k \frac{SF(k)}{speed_k} * \frac{SF(k)}{speed_k}$

$\text{RF} = \text{RF} - \min_k \frac{SF(k)}{speed_k} * \frac{SF(k)}{speed_k}$

Update active links and outgoing speed

end if

end while

end for

C.2 Diverging Nodes

For diverging nodes, there is just one incoming link but multiple outgoing links. Traffic flow is turning to one or more outgoing links. If the one of the outgoing flow is blocked,

the flow on the incoming link is blocked no matter which outgoing link it is turning into. During each time slice, the key is to find a previous time point t_p such that all the vehicles entered the upstream link before t is able to pass through this intersection while all the vehicles entered the upstream link after t_p cannot. Similarly, after getting cumulative departure on incoming links, congestion cost can be easily calculated. For simplicity, free flow travel time is assumed as 0 since they don't affect the increasing/decreasing cost.

Pseudocode:

Assuming 1 incoming links and K outgoing links, Increase the number of lanes on one of the outgoing links by 1 (decreasing is the same), get new outgoing capacity. Calculate maximum receiving flow during Δt by $RF(k) = updated\ capacity(k) * \Delta t$

$C^{up}(k,t)$ is cumulative arrival for incoming link turning into link k.

For t from 1 to $\frac{T}{\Delta t}$:

set $C^{down}(k,t) = C^{down}(k,t-1)$

Update maximum sending flow to each link: $SF(k) = C^{up}(k,t) - C^{down}(k,t-1)$

Find a previous time point $t_p(k)$ for each k, such that $C^{down}(k,t-1) + RF(k)$ entered the upstream link and will turn to link k

$t_p = \min(t, \min_{k} t_p(k))$

Update $C^{down}(k,t) = C^{up}(k,t_p)$

end for