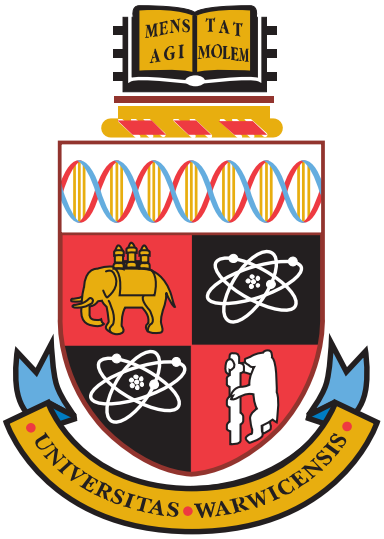


Deep Spatio-Temporal Learning for Dynamic Urban Shared Mobility Systems

by

Man Luo

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

May 2021

THE UNIVERSITY OF WARWICK

Contents

Acknowledgments

I would like to express my sincere gratitude to my PhD supervisor Prof. Hakan Ferhatosmanoglu for the continuous support for my research. His enthusiastic and rigorous attitude for research greatly motivated me. I would like to thank Dr. Hongkai Wen for his research vision and technical depth which inspired me. I also thank the Alan Turing Institute for funding support during my study.

I would also like to thank Professor Abhir Bhalerao, who was my supervisor during undergraduate, by whom I was encouraged to pursue a PhD. I am grateful to Professor Jianfeng Feng, from whom I learned how to conduct impactful and multi-disciplinary research. I thank my colleagues in the group, especially Bowen Du, who gave me great support and valuable discussion. Finally I fell honored to work with Wenzhe Zhang, Tianyou Song and Kun Li who were interns in our group.

I owe my upmost thanks to my family, Mom and Dad. I couldn't come this far without your love and encouragement.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work presented was carried out by the author except where acknowledged. Parts of this thesis have been previously published by the author in the following:

1. **Man Luo**, Bowen Du, Konstantin Klemmer, Hongming Zhu, Hongkai Wen. "Deployment Optimization for Shared e-Mobility Systems with Multi-agent Deep Neural Search." Under revision in *ACM Transactions on Intelligent Transportation Systems (**T-ITS**)*, 2021.

2. **Man Luo**, Wenzhe Zhang, Tianyou Song, Kun Li, Hongming Zhu, Bowen Du, Hongkai Wen. "Rebalancing Expanding EV Sharing Systems with Deep Reinforcement Learning." In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (**IJCAI**)*, pp. 1338-1344, 2020.

3. **Man Luo**, Bowen Du, Konstantin Klemmer, Hongming Zhu, Hakan Ferhatosmanoglu, Hongkai Wen. "D3P: Data-driven Demand Prediction for Fast Expanding Electric Vehicle Sharing Systems." In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (**IMWUT**)*, 4(1): 21:1-21:21, 2020.

Research was performed in collaboration during the development of this thesis, but does not form part of the thesis:

1. Kai Wu, Bowen Du, **Man Luo**, Hongkai Wen, Yiran Shen, Jianfeng Feng.

"Weakly Supervised Brain Lesion Segmentation via Attentional Representation Learning". In *Proceedings of the 22nd International Conference on Medical Image Computing and Computer Assisted Intervention (**MICCAI**)*, pp.211-219, 2019.

2. Bowen Du, Chris Xiaoxuan Lu, Xuan Kan, Kai Wu, **Man Luo**, Jianfeng Hou, Kai Li, Salil S. Kanhere, Yiran Shen, Hongkai Wen. "HydraDoctor: Real-time Liquids Intake Monitoring by Collaborative Sensing." In *Proceedings of the 20th International Conference on Distributed Computing and Networking (**ICDCN**)*, pp. 213-217, 2019.

Parts of this thesis were supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

Abstract

Shared mobility systems have been recently tested and piloted in many cities across the globe, with great potentials to be deeply woven into the fabric of future urban planning. Those systems, not only represent a more sustainable paradigm that can effectively cut unnecessary emissions, but could also bring significant societal benefits by offering a much more affordable on-demand mobility option to the general public. As a relatively new mobility trend, they expand at impressive speeds, pouring more fleet and infrastructure into the urban spaces than ever before. Coupled with the unpredictability of real-world environments, this brings numerous challenges in the deployment and operation of those systems, impacting their usability and practicality. In this thesis, we aim to better understand the structure, process and interaction of the shared mobility system with such dynamics, from a spatio-temporal learning perspective. Particularly, this thesis addresses the following question: *How deep spatio-temporal learning can be tailored to improve the prediction, optimisation and actuation of shared mobility systems in the presence of real-world dynamics?*

The key insight is that such dynamics, although complex, shouldn't be treated independently in an isolated way, but rather be considered and embraced at full stack, from data modelling to inference and learning across the spatio-temporal domain. Specifically, we tackle this in a number of research threads. Firstly, we propose $\mathbf{D}^3\mathbf{P}$, a novel data-driven prediction framework, which is able to directly model the spatio-temporal dynamics from the data with time-varying graphs, and uses bespoke dynamic Graph Convolutional Neural Networks (GCNs) to accurately forecast

the future demand of the shared mobility systems in both short and long terms. We further propose **MANS**, a new optimisation approach for mobility infrastructure deployment based on multi-agent neural search, which can effectively discover the optimal infrastructure deployment strategies for shared mobility systems across space and time, so that the services provided are ubiquitous to the users while sustainable in profitability. Finally, we propose **ac-PPO**, a novel user-incentive based fleet management approach, which uses a deep reinforcement learning paradigm to guide the rebalancing of fleet with cooperative users, and incorporates the dynamics by a new action cascading technique. All the proposed approaches have been comprehensively evaluated through large-scale experiments with extensive real-world datasets, and the results have shown superior performance compared with the state-of-the-art, demonstrating their potential impact on a broad spectrum of application scenarios.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

One of the most significant trends of future urban mobility is sharing. According to a recent study [1], by 2050 the average time that an urban travellers spends in traffic jams could be 106 hours per year, three times more than today. This motivates a massive surge of investments in the public transportation sectors across the globe, resulting in up to two-fold increase of market volume predicted by the International Association of Public Transport [1]. However, while public transport remains a fundamental form of urban mobility, which in fact is also shared, the emerging challenges of future urban mobility may not be fully addressed by public transport only. Driven by digital platforms and new technologies, new mobility models and systems based on shared ownership, use and access of infrastructure (including bikes, vehicles, chargers, etc.) are proliferating. We are witnessing a major change in users' preferences towards urban mobility, embracing novel approaches to balance individuality and sharing, which is also in line with the recent shift towards sharing economy in other sectors, such as accommodation, finance and learning [2].

The benefits of developing such shared mobility paradigm in urban spaces are clearly visible. First of all, it is a greener mobility option which can effectively cut unnecessary carbon emissions on the roads, especially when clean energy is used, e.g. electric vehicles/scooters. With the recent advances in battery technologies, this electrified form of shared mobility has been deployed in various major cities, from London [3], to Berlin [4] and Singapore [5]. Secondly, the sharing nature of such systems, in that multiple users undertake trips with the same mobility resources, or companies share costs for transport of goods, could make them more accessible and affordable than individual ownership. In practice, the implementation of this shared

access based on users' demand disconnects the usage of mobility infrastructure from their ownership, further maximising the utilisation of total resources that our society can afford. Finally, compared with traditional solutions, shared mobility systems are often more efficient and flexible, since they essentially exploit the idle time of urban mobility resources (e.g. according to [6], in the UK privately owned cars are unused 95% of the time), which are considered as the "wasted capacity", opening up new opportunities in the urban mobility space. In addition, thanks to the adoption of new technologies in sensing, data acquisition and scheduling, share mobility systems are becoming more data-centric and intelligent, capable of providing the ever better travel experience: e.g. people can hail a taxi or source a bike at just a few clicks in an app on their mobile phones, almost everywhere in the urban environment.

Besides improving their own services, the rich data generated by shared mobility systems can also benefit a broader spectrum of urban applications, from traffic management, to city planning and facility development. In light of this, many organisations including governments and major companies in the shared mobility sector have been actively engaged in open data initiatives, such as the London Datastore [7], NYC Open Data [8] and the GAIA project [9], advocating transparent access to the relevant information. These data, typically in the format of spatio-temporal records, such as trajectories, origin/destination pairs, and infrastructure deployment processes, contains key insights on how such shared mobility systems behave in the real-world urban environments across space and time. It has been shown by many existing studies [10, 11] that leveraging the data generated by shared urban mobility systems can enable greater innovation, realise more efficient services, and support resilient communities.

Therefore, the problem investigated in this thesis is how to better *model*, *optimise* and *actuate* the shared mobility systems in urban spaces, where a variety of entities such as users, fleets and mobility infrastructure interact with each others and generate heterogeneous data footprints across spatial and temporal domains. In particular, we study the shared mobility systems in *dynamic* settings, in that they may evolve over time, e.g. varying their operations or infrastructure under different circumstances.

1.2 Challenges

We have identified the following key challenges in addressing the above problem:

Firstly, in reality shared mobility systems are operated in urban environments with unpredictable real-world dynamics. This means those systems can be

heavily affected by various factors, such as weather, events, holidays, and other transportation modalities, making accurate modelling and prediction of their behaviour challenging. In addition, as a relatively new mobility paradigm, in many cities the shared mobility systems themselves are continuously expanding their service/infrastructure at visible speeds. As shown later in this thesis, this expansion process introduces significant extra dynamics, leading to new challenges on the modelling, optimisation and actuation of those systems that haven't been identified nor fully addressed by existing approaches.

Secondly, in practice it is often not economical or even feasible to actuate/optimise the shared mobility systems *in vivo*, e.g., implementing different operation strategies and retrospectively evaluating their corresponding effects, especially across the entire city. In this case, a common practice is to adopt simulation [12, 13]. However, building a high-fidelity simulation environment for shared mobility systems at urban scale can be challenging, in that i) it is not trivial to faithfully capture the fine-grained interactions between various elements involved in shared mobility systems, such as users, fleets and infrastructure; ii) calibration of the simulation environment are not straightforward, which often requires careful design to incorporate multi-modal real-world data; and iii) balancing simulation performance and efficiency can be difficult, especially when the system operates across large ubran spaces, involving substantial amount of fleet and infrastructure.

Thirdly, the data footprint generated by shared mobility systems can be imprecise, incomplete and of huge volumes. This poses significant challenges in processing and modelling the data, requiring novel machine learning approaches that can scale gracefully on large data volumes (e.g. taxi trajectory data in a single day could have millions of data points), and are resilient to various types of imperfections (e.g. data loss caused by sensor malfunctioning, network faults etc.). In particular, the fact that shared mobility systems are continuously expanding exacerbates these challenges, e.g., the systems may expand to new areas with limited prior knowledge or historical data, making it very challenging to effectively model or optimise the systems based on information available.

Finally, effective learning for the spatio-temporal data produced by shared mobility systems is a challenging task. Although deep learning methods have demonstrated superior performances in many applications, such as image recognition and speech processing, most of the existing techniques are designed either primarily for spatial input, e.g. images, or temporal input, e.g. audio signals. In our case, the spatio-temporal data generated by shared mobility systems has both spatial and temporal properties that are much more complex, which can't be directly cap-

tured by off-the-shelf learning techniques. In addition, one common assumption in standard learning approaches is that the underlying data samples are generated independently (e.g. governed by i.i.d. variables). However, for spatio-temporal data which encodes non-trivial dependencies within and across the spatial and temporal domains, this is often not true: the data tends to be highly self correlated, posing additional challenges for modelling and learning.

1.3 Contributions

To tackle these challenges, in this dissertation, we developed new modelling and learning techniques for urban shared mobility systems, aiming to better understand and optimise such systems in the presence of real-world dynamics. To this end, it presents a class of novel approaches in a variety of scenarios, from demand forecasting to infrastructure optimisation and fleet management. Concretely, the technical contributions of this thesis are:

Accurate prediction of mobility demand is crucial for shared mobility systems to operate effectively and efficiently, especially when the systems are continuously expanding. In this thesis we propose a novel data-driven approach called D^3P for demand prediction, which is capable of modelling the complex dynamics caused by such system expansion. In particular, we model the expanding shared mobility system as time-varying graphs, and consider both temporal dependencies and spatial correlations within the systems by employing a novel dynamic Graph Convolutional Neural Network (GCN). On top of that, we further design a new multi-scale predictor, which is able to not only forecast the expected future demand of the system, but also the concrete instance demand during a given time period (e.g. specific days in the near future), allowing the stake holders to better understand both short and long term performance of the systems. Extensive experiments on real-world data show that the proposed D^3P approach significantly outperforms the state-of-the-art forecasting approaches, offering up to three-fold improvement in prediction accuracy and is robust to different levels of expansion dynamics.

High-fidelity simulation of the shared mobility systems is the cornerstone for many tasks that require learning via trail-and-error, such as infrastructure optimisation and fleet management. To support development of such a simulation environment, we worked with an industry partner to conduct an in-depth case study of a real-world shared mobility system for one year, and collected rich sets of data from various sources to support our study. We analyse the operational models, expansion processes and usage patterns of the system, and abstract key functionalities and

operations that are crucial. We further show how to calibrate the simulation environment with data collected from the real world, ensuring it to faithfully capture the fine-grained details when the system interacts with users and infrastructure. To support reproduciblility, we've also open-sourced the code [14] of our simulator to the community.

Infrastructure optimisation plays an important role in the deployment and expansion of urban shared mobility systems. Existing solutions often assume that the optimisation may only happen once for all, while in this thesis we consider a more realistic scenario, where the deployment of infrastructure could be optimised dynamically as they operate. In particular, we would like to find the dynamic deployment plan that guides system deployment through space and time, which is optimal in terms of given objectives (e.g. service coverage or revenue) and constrains (e.g. different stages of budget). To address that, we design a multi-agent neural search algorithm called MANS, which works with our simulation environment and uses reinforcement learning to discover viable deployment plans in spatio-temporal domains. Specifically, we develop a hierarchical controller for the search process, which iteratively proposes deployment plans and evaluates their performance. The results are then propagated back to the controller as reward signals, whose parameters are updated accordingly so it can generate better plans in future iterations. Through extensive evaluation, we show that development plans proposed by our approach significantly outperform the actual plans used in the real-world, and the existing one-stage or multi-stage optimisation approaches, achieving improvements in both revenue and service coverage.

Fleets are the lifeblood of any urban mobility system, where for shared mobility systems, effective fleet management (e.g., rebalancing) is the key enabler to unlock their full business potentials, especially when the systems are continuously expanding. To the best of our knowledge, this thesis is the first to identify the problem of rebalancing the expanding shared mobility systems. We consider the user incentive-based rebalancing paradigm, and formulate the rebalancing problem under the multi-agent reinforcement learning (MARL) framework, where we design the agents, states and rewards for the specific context accordingly. To solve that, we propose a novel approach of policy optimisation with action cascading called ac-PPO, which uses two connected policy networks to handle the dynamics introduced by the continuous expansion of the shared mobility systems. We also design a regularised reward function for the proposed ac-PPO, which can effectively stabilise training and improve data efficiency. With our simulation environment, the proposed ac-PPO approach has been evaluated extensively, and results show that

our approach significantly outperforms the state-of-the-art, offering significant improvement in system net revenue and user demand satisfied rate.

1.4 Thesis Structure

The rest of this thesis is organised as follows. Chapter 2 covers the literature survey for the existing methods and work, which provides a general background for the spatio-temporal learning. The following three chapters present the proposed approaches of this thesis. Chapter 3 introduces the proposed data-driven demand forecasting approach D^3P for accurate prediction of mobility demand in the presence of real-world expansion dynamics, which tackled the first challenges identified above. Chapter 4 presents the design of our simulation environment, and the proposed infrastructure optimisation approach MANS, which uses multi-agent neural search to discover optimal deployment strategies. Specifically, we introduce a hierarchical controller, which iteratively proposes deployment plans, and evaluates their performance and then the results are propagated back to the controller as rewards, whose parameters are updated accordingly so it can generate better plans in the future. Chapter 5 discusses the proposed fleet management approach ac-PPO, which formulates the task as a Multi-Agent Reinforcement Learning (MARL) problem and introduce techniques of action cascading for policy optimization, which is able to work with the dynamics in the system and solve the imbalance problem via incentivizing users. Finally in Chapter 6 we conclude the thesis and outline potential future directions. Now we are in the position to go through the literature related to this thesis.

Chapter 2

Literature Review

This chapter covers the background and existing work related to the contributions of this thesis. Specifically, in Section 2.1 we briefly introduce the concept and taxonomy of shared-mobility systems in urban settings, which describes the main environment where all the proposed approaches in this thesis are working on. Then in Section 2.2 we survey existing machine learning techniques for spatio-temporal data analytics, from the traditional "shallow" learning methods to more recent deep neural networks (DNNs) and geometric deep learning approaches, where the traditional "shallow" learning methods and deep neural networks are used as baselines in the experimental parts of the following contribution chapters, while the geometric deep learning approaches are core components for the proposed demand forecasting approach in Chapter 3. Finally in Section 2.3, we discuss reinforcement learning techniques, another important research field closely related to this thesis, and highlight their applications in the urban mobility sector as both of the infrastructure optimisation in Chapter 4 and fleet management in Chapter 5 are solved through reinforcement learning approaches.

2.1 Shared Mobility Systems in Urban Settings

Broadly speaking, urban mobility refers to the trips generated daily by the inhabitants from one point in a city to another, which are associated with methods, conditions and data, such as the modes of transport selected, length of trip, time spent in transport, etc [15]. In the early ages, the trips generated by people were far less due to the limitation of travel methods. With technology advances, numerous new modalities for travel have been invented, such as buses, subway, or private cars. However, the increase in urban population poses significant challenges to the

capacity of transport infrastructure of the cities, where now most of the world's cities are congested and heavily polluted. In addition, new economy models such as e-commerce introduces extra demand for mobility, where people and goods flow across urban areas, making mobility a key competency for modern cities to function. To alleviate those challenges, shared mobility systems, which have already modified our travel habits during the past decades, emerged with the aim to improve the utilisation of urban mobility resources and achieve more flexible ways of travel. In the following, we start by revisiting the traditional shared mobility systems such as buses and taxis, and then discuss the more recent ride-hailing and shared bike/vehicle systems that have been increasingly popular in major cities across the world.

2.1.1 Traditional Urban Shared Mobility Systems

Generally, the term "shared mobility" refers to the modality where transportation services and resources are shared among users, either concurrently or sequentially, i.e. one after another. In that sense, buses, taxis and other public transportation systems, which are based on regular operation along fixed routes according to certain timetables or arbitrarily across the urban environments, can be categorised as a form of shared mobility systems. People can access them at locations where they are available such as bus stops or taxi stations, and then proceed to their destination. For many years, these public shared mobility systems are one of the primary transportation modes in cities [16], and have attracted numerous research interests to address the associated challenges, such as bus route planning and taxi dispatching. For instance, the work in [17] proposes a two-phase approach for night bus route planning, aiming to find a bidirectional route that can maximize the number of passengers expected along the route, while the work in [18] designs a dynamic route planning approach for shuttle buses. For taxi, early work in [19] introduces a multi-agent approach for automating taxi dispatching, and later work in [20] develops a robust data-driven taxi dispatching framework, which takes the spatial-temporally correlated demand uncertainties into account. The work in [21] also proposes a passenger finding and driver dispatching strategy using spatio-temporal features, while different service strategies of taxi drivers have been studied in [22]. To better understand the behaviour of such systems, historical GPS traces of vehicle trajectories are often considered, e.g., for optimal route selection [23].

2.1.2 Ride-hailing Systems

Recently, in addition to the traditional taxi services, a new form of shared mobility systems, named ride-hailing services, such as Lyft, Uber and Didi [24] are gaining their popularity in major cities. Those systems, unlike the standard taxi which have to be hailed from streets or taxi ranks, allow users to order a customised ride using smartphone apps to travel from A to B [24]. In this way, ride-hailing systems offer more comfortable on-demand mobility options for door-to-door travel, and are usually cheaper than their counterparts [25], leading to increase in adoption among urban travellers. However, compared to traditional shared mobility systems, the ride-hailing systems, as an emerging modality the ride-hailing systems, also introduce new challenges. One fundamental problem that has attracted a lot of research efforts is how to effectively and efficiently match the mobility requests of the users with the current states of the system, e.g. capacity of nearby available drivers. For instance, the work in [26] develops an order dispatching algorithm for such ride-hailing platforms, aiming to optimise the platform's long-term efficiency, while in [27] the authors consider preference-aware order assignment using online stable matching to maximise the expected total profit, as well as minimising the expected total number of blocking pairs. Another parallel yet important problem for the ride-hailing systems is the design of an optimal pricing strategy, i.e., when and how much to charge the users to trade-off maximum profit for customer satisfaction. This can be addressed by dynamic pricing, e.g., the work in [28] proposes the ADAPT-Pricing scheme, which sets prices based on different origin destination pairs of the rides rather than the origins only. Besides profitability, achieving certain levels of fairness in pricing strategies is also essential to the success of such ride-hailing systems. For instance, the work in [29] introduces a fair pricing model, which seeks for profit while satisfying constraints (e.g. availability and affordability) of different user groups.

2.1.3 Station-based Ride-sharing Systems

Unlike the ride-hailing systems where users typically won't need to drive the vehicles by themselves, recently, another shared-mobility paradigm, station-based ride-sharing systems have been widely adopted in different cities, e.g., shared bikes including Capital Bikeshare in Washington DC [30] and Velib in Paris [31], and shared cars or car clubs such as Zipcar in London [32]. Conceptually, a station-based ride-sharing system maintains a set of designated stations (in some cases can be just parking spaces) across the city, where each station has a fixed capacity for fleet

to park, i.e., bikes or cars. Using the systems is straightforward: a user can pick up a bike/car from any stations to start the trip, and then return the bike/car to another station when finish. The rental price is typically calculated based on trip duration. Since those systems depend heavily on their infrastructure, i.e., stations, one natural problem arises is that how stations should be deployed across the city to improve their operations. The work in [33] tackles this problem by leveraging open data to forecast potential trip demand, using a two-phase feature selection method to extract customised features for prediction. On the other hand, the work in [34] proposes a station optimisation approach for bike sharing systems, which selects the appropriate station locations by integrating multiple factors into decision process. Another common problem for such station-based ride-sharing systems is the imbalance distribution of their fleets across the stations. Intuitively, for those systems to work well, at most of the time, a station should have both sufficient number of vehicles for users to borrow, and ample amount of parking/docking spaces for users to return. However, due to certain patterns of human mobility, e.g., commuting to central areas in the morning, the fleet distribution often becomes very skewed, and thus rebalancing is needed. For example, in order to rebalance bikes with minimum cost, the work in [35] develops different bike reservation policies, and suggests users to visit the less crowded stations instead of the busy few. Another line of work focuses on minimising the rebalancing cost in terms of distance travelled, e.g., in [36] a heuristic method is developed for single vehicle static rebalancing, which solves the rebalancing problem for up to 60 stations with a branch-and-cut procedure. The work in [37], on the other hand, incorporates data from multiple sources to optimise rebalancing in bike sharing systems.

2.1.4 Station-less Ride-sharing Systems

Unlike the station-based ride-sharing systems which needs stations to park/dock vehicles/bikes, recently the station-less ride-sharing systems are becoming popular, particularly in the shard bike sector, such as MoBike and ofo [38]. Those systems relax the requirements that shared bikes should only be rented from and returned to designated stations, but the users can take any available bike on the street to start the trip, and return to any locations that are compliant with urban regulations. Intuitively, such station-less systems are able to provide more flexible and convenient travel experiences than their station-based counterparts, but they may suffer more from the problem of imbalanced fleet. Since there is little or no restrictions on where the bikes can be parked, in some cases the bikes may be piled at certain locations, causing serious problems such as blockages along road segments. Recent work has

identified and tried to mitigate such problems. For instance, the work in [39] detects possible parking hotspots before the station-less ride-sharing systems were deployed by fusing multi-source urban data, allowing appropriate actions to be taken a priori to avoid potential pitfalls. The work in [40] proposes a pricing scheme to incentivize users to voluntarily park the bikes to less congested locations, i.e. rebalancing the system. On the other hand, the fact that those systems are station-less means that the trajectories generated by them can be richer and more diverse, which may embed useful information for tasks such as urban planning and management. For example, the work in [41] leverages trajectory data from real-world station-less bike sharing systems to help the planning of new bike lanes in cities, while the work in [42] uses similar information to detect illegal parking events in urban environments.

2.1.5 Shared Electric Mobility Systems and Infrastructure

With the recent advances in battery technologies and the increasing concerns in emissions, Electric Vehicles (EVs) have gained rapid adoptions in many urban mobility systems. However, this electrification of fleets has introduced a number of new challenges. Unlike the traditional internal combustion engine (ICE) powered vehicles, the EVs typically have limited range, while the charging time can be much longer than filling up the petrol or diesel cars. In addition, although growing continuously, the charging infrastructure for EVs is still less ubiquitous than petrol stations, where the potential incompatibility between chargers and battery specifications makes it harder to charge a EV smoothly [43]. In the context of shared mobility systems, such unique properties of EVs may also set constraints in their normal operations, e.g. EVs can't be driven beyond the remaining range without charging, and they often need to be sufficiently charge to serve future orders. The work in [44] studies these properties in the context of electric taxi in a city, which conducts a comprehensive measurement investigation on the long-term evolving mobility and charging patterns of those systems, using real-world data collected over five years. The work in [45] considers route planning and optimisation for EV fleet, which uses a multi-objective route planner to guide the EVs. There is also a rich body of work [46, 47] on EV charging scheduling, which studies how to plan for different types of EVs to be charged efficiently across space and time. Another active research area is the development and planning for EV charging infrastructure [45]. For instance, the work in [48] considers region-based charger planning to minimise range anxiety of EV users, while the work in [49], the interactions between charger deployment and EV users are taken into account when planning the infrastructure. The work in [10] proposes a charger planning approach which aims to maximise

the overall satisfied user charging demand with given budget constraints, while the work in [50] considers a heuristic algorithm to also take the social factors (e.g., fairness) into account when deploying charging facilities, arguing that charging demand should be satisfied not only in central places but also in relatively rural areas.

2.2 Machine Learning for Spatio-temporal Analysis

After reviewing the concepts and taxonomy of shared-mobility systems in urban settings, and it is obvious that the majorities of entities in our cities operate and evolve across both space and time, forming spatio-temporal processes that describe the dynamics and interactions between them, e.g. the moving of vehicles forms traffic flow, and the development of infrastructure reshapes functionalities of different urban regions. With the recent development of sensing and wireless communication technologies, capturing and streaming data from those processes becomes much more straightforward, enabling a number of new applications such as intelligent traffic management, mobility modelling and urban climate control. Those applications, although with different objectives and assumptions, rely heavily on the capability of extracting relevant knowledge from the collected spatio-temporal data, a task often known as spatio-temporal analysis. Machine learning techniques, on the other hand, have demonstrated superior performance in processing data in both spatial and temporal formats, from classifying objects in images to recognising speeches with audio signals. These techniques and many of their variations, have also been used to analyse the spatio-temporal data, particularly in urban computing context, e.g., improving service levels of taxi [51], predicting user demand [52] and repositioning vehicles in shared mobility systems [40]. In the following, we thoroughly survey the machine learning approaches that are relevant to spatio-temporal analysis, from the traditional shallow learning approaches, to the more recent deep learning and reinforcement learning.

2.2.1 "Shallow" Learning Approaches

With slight abuse of terms, we use "shallow" learning to refer to the conventional machine learning models, which do not use deep neural networks. For spatio-temporal data which contains correlations along both spatial and temporal axis and between them, the shallow learning approaches typically model the dependencies in spatial and temporal domains separately or in tandem, e.g., they may extract spatial features (e.g., geographical distance) and temporal properties (e.g., periodical patterns) from the data, and use those features for learning. Therefore without loss of gener-

ality, in the following we will cover the existing shallow methods on spatial learning and temporal learning respectively, where those methods are often employed jointly for spatio-temporal analysis.

Spatial Learning. Traditional approaches for spatial learning typically consider clustering and kernels, which aims to group features in the spatial domain, or interpolate based on existing data samples. A simple yet widely used clustering algorithm in spatial learning is k-means, which firstly identifies k centroids and then assigns each data point to the clusters represented by the nearest centroid. For instance, the work in [53] uses k-means to identify the spatial pattern in storms and [54] considers a similar clustering approach to profile hotspots of road accident. Other popular clustering algorithms for spatial learning include hierarchical clustering and density-based approaches such as DBSCAN [55], which uses different techniques to extract correlations of the data in spatial domain. A similar but slightly different learning approach often used in classification and prediction tasks for spatial data is the Supported Vector Machine (SVM), which aims to find a hyperplane in feature space that can best separate the data points [56]. For example, the work in [57] applies SVM in environmental data classification and [58] proposes an algorithm for short-term traffic flow prediction using adaptive SVM. For interpolation in spatial data, Gaussian Processes (GPs) [59] are popular models, which are able to directly represent uncertainty and thus can be used to gain physical insight into the processes. Also known as Kriging [60], GPs have been widely considered in various interpolation/regression tasks in spatial learning, particularly in the urban computing/GIS community, such as air quality estimation [61], precipitation forecasting [62] and demographic analysis [63].

Temporal Learning: Learning approaches for temporal data (e.g., time series) primarily focus on prediction/forecasting tasks. Single Exponential Smoothing (SES) and Auto Regressive Integrated Moving Average (ARIMA) are the two most widely used approaches, where the former is often used for data with no clear trend or seasonal pattern while the latter aims to capture the autocorrelations in the data. Concretely, SES requires a single parameter, called the smoothing factor, which is often set to values between 0 and 1. Larger values indicate that the more recent data points would have bigger impact on the model, while smaller values mean the contrary [64]. Using the SES model, the work in [65] forecasts urban water ecological footprints using exponential smoothing analysis and in [66] exponential smoothing is adopted for urban traffic forecasting. On the other hand, ARIMA is often parameterised as ARIMA(p, d, q), where p is the order of autoregressive model (i.e., number of time lags), d is the degree of differencing (i.e., the number of times the data have

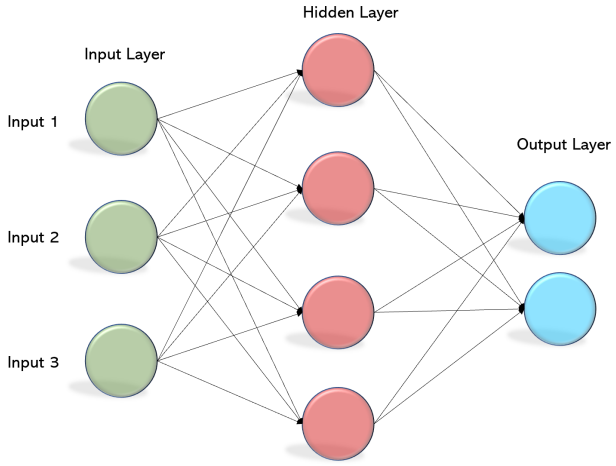


Figure 2.1: A minimalist example of the multi-layer perceptron (MLP).

had past values subtracted) and q is the order of moving-average model [67]. Those parameters can be adjusted to fit the data as well as possible. ARIMA has also been applied in many urban application scenarios that involve time series analysis, e.g., the work in [68] adapts ARIMA to forecast daily mean ambient air pollution, while in [69] a variant of ARIMA model has been considered for travel time prediction in urban environments.

2.2.2 Deep Neural Networks

Deep learning methods have gained significant popularity thanks to their superior performance in a broad range of applications, such as object detection [70, 71], speech recognition [72, 73], natural language processing [74], machine translation [75] and many other areas [76, 77]. Essentially, they are very good at approximating complex non-linear functions and capable of extracting features automatically from raw data without the effort of pre-processing and hand-crafted feature engineering [78]. Therefore, deep learning approaches have also been introduced to spatio-temporal analysis, and demonstrated promising results in various application scenarios. In the following, we will give a brief overview of the different types of deep neural networks that are considered or relevant to the approaches proposed in this thesis, including the multi-layer perceptron (MLP), convolutional neural networks (CNNs), recurrent neural networks (RNNs) and graph neural networks (GNNs).

Multilayer Perceptron (MLP). A MLP is a class of feedforward Artificial Neural Networks (ANNs), which typically consists of at least three layers of neurons: an input layer, a hidden layer and an output layer. Each of those layers contains neu-

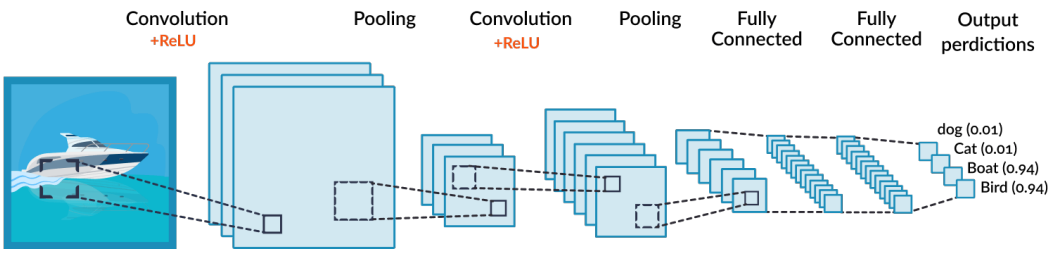


Figure 2.2: An example of Convolutional Neural Networks (CNNs) used for image classification task.

rons, which are wired by connections. Neurons receive inputs from their predecessor neurons in previous layer and produce output to the successor neurons in the next layer, via certain activation functions, which can be either linear or nonlinear. Note that there is no connections between neurons in the same layer. The connections between the neurons are associated with weights, which are multiplied to the input signal. Therefore, the output of each neuron is a weighted sum of the inputs followed by neuron's activation function, which is usually nonlinear. Fig. 2.1 shows a minimalist example of a MLP.

The process of adjusting the weights during the learning process is often refereed to as training. In practice, the most common training technique for MLP and other ANNs is gradient descent. It calculates the gradient of each weight with respect to a predefined loss function, which measures the difference between the output and the ground truth (i.e. labels). Essentially, it estimates by what amount the error will increase or decrease if the weight were increased by a tiny amount. To calculate the gradients, a standard approach is *back-propagation*. The idea is to first perform a feed-forward pass from the input neurons to the output neurons, and compute the loss between the output and the labels. Then a backward pass is performed, where given the loss, the gradient of each weight can be iteratively calculated according to the chain rule. The weights are adjusted in the negative direction of the gradient, where the step size of the adjustments is proportional to the magnitude of the gradient and the *learning rate*, a hyperparameter defined a priori.

Convolutional Neural Networks (CNNs). In deep learning, CNNs are designed particularly to be shift/space invariant, which have been commonly applied to process image data. It takes the advantage of the properties of natural signal: local connections and shared weights, where the convolution kernels (i.e., filters) slide along the input data/features and output translation equivariant responses (often known as the feature maps). The weights sharing mechanism vastly reduces the

number of parameters compared to the fully connected MLP with the same input and output dimensions. Fig. 2.2 shows a typical CNN architecture for image classification, which contains three types of layers: the *convolutional layer*, the *pooling layer* and the *fully-connected layer*.

- A convolutional layer convolves the input and output the result to the next layer, using convolutional filters and non-linear activation functions. It mimics the response of a neuron in the visual cortex with respect a specific stimulus, where each neuron only processes data for its receptive field. Compared to the fully connected layers used in MLPs, the convolutional layers require much fewer learnable parameters due to weight sharing between filters, allowing deeper networks in practice. It is particularly suitable for processing data with a grid-like topology (e.g., images) since spatial correlations are naturally considered during convolution.

- A pooling layer performs downsampling along the spatial domain, which combines the outputs of neuron clusters at one layer into a single neuron in the next layer. In practice, pooling can be either local or global, where local pooling operation considers neurons in small clusters, where global pooling operates on all neurons of the feature map. There are two types of pooling operation commonly used, max pooling and average pooling. The former extracts the maximum value of the cluster or feature map, while the latter computes the average value.

- A fully-connected layer is the same as the layers in MLP as discussed above, where every neurons in the previous layer is connected to every neuron in this layer. In CNNs, it is often used to flatten the output from the last convolutional/pooling layer, to generate the predicted labels.

Essentially in a typical CNN, the convolutional layers perform non-linear transformations to the input features, where spatial correlations are taken into account by the convolution operations, while the pooling layers reduce the feature dimensions. Similar to the MLPs, CNNs can be trained using the family of gradient descent algorithms. In practice, CNNs have demonstrated their success in a number of different computer vision tasks, from object detection [79] to robotics perception [80] and autonomous driving [81]. In the urban computing community, CNNs have also gained considerable interests and been applied to a number of tasks that require capturing the complex spatial correlations, such as citywide crowd flow prediction [82] and traffic speed estimation [83].

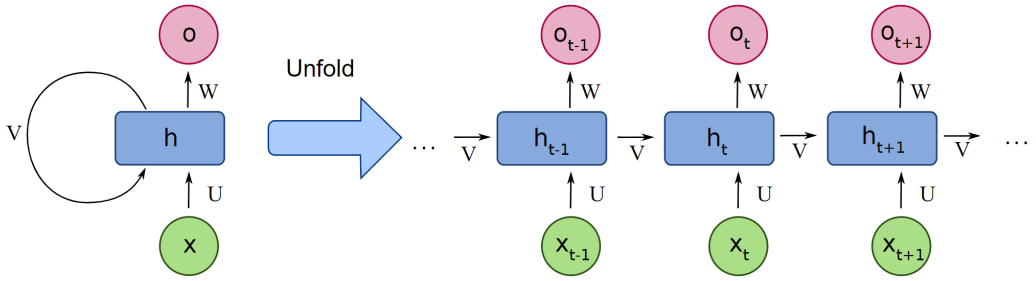


Figure 2.3: An example of (left) Recurrent Neural Networks (RNNs) and (right) its unfolded form in time.

Recurrent Neural Networks (RNNs). RNNs are neural networks designed for processing sequential or time series data such as audio and text, which consume one element of the input sequences at a time, while maintaining their hidden states in a "state vector" that implicitly contains information about the history of all past elements of the sequence. Fig. 2.3 illustrates an example of the typical RNNs and the associated unfolded form in time, where x is the input and o is the output. h is the hidden state of the RNN, which maintains the internal "memory" of the input sequence. U, V and W are the weights of the RNN, which are shared among all the timestamps. Therefore, a RNN can map an input sequence $\boldsymbol{x} = \{x_t\}$ to an output sequence $\boldsymbol{o} = \{o_t\}$, where each o_t depends on all the previous input $\{x_1, ..., x_t\}$. Given its unfolded form (As shown on the right of Fig. 2.3), it is straightforward to train a RNN using backpropagation: we could consider the unfolded network as a very deep feedforward network, where the outputs at different timestamps in the RNN are the outputs of different neurons in the feedforward network, and all layers share the same weights.

Although theoretically RNNs can process arbitrarily long input sequences, in practice, it has been shown that it is difficult to remember long-term dependencies over time due to the vanishing gradient problem [84]. Long short-term memory (LSTM) networks [85] are one of the most popular RNN variants that can address such problems, which use a special hidden unit called the memory cell to handle information from the past. In particular, a common LSTM unit also contains an input gate, an output gate and a forget gate. Those gates regulate the flow of information (i.e., features in forward pass while gradient in backward pass) into and out of the cell, so that the cell can remember states over arbitrary time. LSTMs can partially avoid the gradient vanishing problem, as they allow gradient to flow unchanged across the network with the help of those gates. Due to such desirable

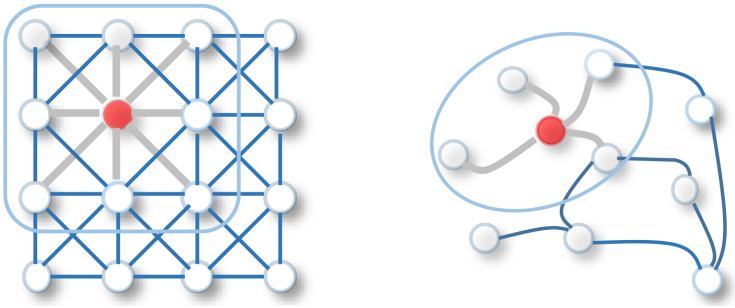


Figure 2.4: (Left) Standard 2D convolution on Euclidean data. (Right) Convolution on graph structures.

properties, LSTMs (or RNNs in general) have been widely applied in applications such as language modelling [86] and machine translation [87], while recently they have also been used extensively in the urban computing context where time series analysis plays an important role, e.g., estimating mobility flow [88], predicting traffic condition [89] and detecting travel mode [90].

2.2.3 Graph Neural Networks (GNNs)

As we discussed above, deep neural networks, including MLP, CNNs and RNNs, have demonstrated unprecedented performance in various tasks involving different types of data, from images to text and audio signals. Here an implicit assumption that those approaches made is that the underlying data has to be *Euclidean*. For instance, images are instances in the 2D Euclidean space, where text and speech data can be viewed as 1D Euclidean structure. However in practice, data can also be *non-Euclidean*: e.g., social networks, gene/protein data and 3D mesh/graphs. For those types of data, it is indeed possible to enforce Euclidean constraints and then adopt standard learning approaches in Euclidean domains, but such conversion often oversimplified the non-Euclidean correlations within the data, and thus harming performance. To address such limitations, Graph Neural Networks (GNNs) are designed to perform inference and learning on non-Euclidean data, e.g., data that can be described in graphs [91]. In the following, we focus on an important type of GNNs, the Convolutional Graph Neural Networks (ConvGNNs), which works in a similar way with the CNNs but on graphs. Fig. 2.4 shows an example of comparison between the standard 2D convolution on Euclidean data such as images (left), and the convolution on graph structures (right). Like CNNs, the intuition behind ConvGNNs is to generate the representation of a given node by aggregating its own features and information from neighbouring nodes (i.e., those nodes connected to

itself via edges). This is also the concept of node embedding in graph theory: we try to map the nodes to an embedding space, in such a space similar nodes in the graph (e.g., carrying similar features or topological properties) are closer to each other. There are mainly two types of ConvGNNs: the *spectral-based* ConvGNNs, and the *spatial-based* GCNs.

Spectral-based ConvGNNs. The spectral-based ConvGNNs are mathematically based on graph signal processing, which generalise the convolution operation to graph structures. In particular, it considers signal processing functions such as the Fourier transform, tools usually reserved for signals/frequencies, and applies them to graphs. Under this setting, the convolution operations are performed by finding the eigendecomposition (a way of factoring a matrix into a set of eigenvectors and eigenvalues) of the graph Laplacian, which is called the spectral decomposition. In fact, the eigenvectors of the graph Laplacian represent the Fourier basis of the input graph, and therefore the convolution on graph can be defined as the multiplication of the input signal $\mathbf{x} \in \mathbb{R}^N$ (a scalar for each of the N nodes) with a filter $\mathbf{g}_\theta = \text{diag}(\theta)$ parameterised by θ:

$$\boldsymbol{g}_\theta * \boldsymbol{x} = \boldsymbol{U}\boldsymbol{g}_\theta \boldsymbol{U}^T \boldsymbol{x} \tag{2.1}$$

where $\boldsymbol{U}$ is the matrix of eigenvectors of the normalised graph Laplacian $\boldsymbol{L} = \boldsymbol{I}_N - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. Here $\boldsymbol{D}$ is the degree matrix of the graph, $\boldsymbol{A}$ is the adjacency matrix, and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues. In practice, most of the spectral-based ConvGNNs follow the above definition, and the key difference between different variants lies in the choice of the filter $\mathbf{g}_\theta$, e.g., the Spectral Graph Convolutional Neural Network (Spectral GNN) [92] considers $\mathbf{g}_\theta$ as a set of learnable parameters, while the ChebNet [93] uses Chebyshev polynomials of the diagonal matrix of eigenvalues to approximate $\mathbf{g}_\theta$, reducing the computational complexity.

Spatial-based ConvGNNs. Unlike the spectral-based approaches that rely on spectrum analysis on graphs, the spatial-based ConvGNNs define convolution operation based on the spatial relations between the nodes, which is very similar to the standard CNNs on images. In that sense, images can be viewed as a special form of graphs, where each pixel is a node, directly connecting to the nearby nodes. A convolutional filter is applied by taking the weighted average of pixel values of the central node and its neighbors across each channel. Similarly, spatial-based ConvGNNs also convolve the central node's representations (features) with the representations of the neighbouring nodes to derive the updated features for that central node. Essentially this allows information propagate across nodes in the graph via edges, and the main challenge is to define convolution operations that work well with differently sized neighbourhoods and thus are able to maintain the local invariant properties like

CNNs. In fact, some popular spatial-based ConvGNNs, e.g., NN4G [94], share very similar filter formulations with the spectral-based approaches (e.g., GCN [95]), but they use the unnormalised adjacency matrix instead of the normalised matrix.

In the context of urban computing, due to their non-Euclidean nature, many real-world problems, such as forecasting, estimation, and inference of demand/traffic/air quality, that require analysis in the spatio-temporal domains, have been tackled with the emerging graph-based deep learning techniques [96, 97]. In those work, spatial correlations are often captured by variants of GNNs as discussed above, while temporal dependencies are typically modelled with certain forms of the recurrent neural networks (RNNs). For instance, the work in [96] models traffic flow as a diffusion process on directed graphs for traffic forecasting, while the work in [52] and [97] propose frameworks that use multi-graph convolutional neural networks to predict demand for taxi and ride-hailing services respectively. Another work in [11] uses an encoder-decoder structure on top of Multi-Graph Convolutional Networks to estimate bike flow between stations within a bike sharing system.

2.3 Reinforcement Learning for Urban Mobility

Besides the machine learning techniques discussed above, reinforcement learning is another popular way to solve many problems in urban settings. Unlike the aforementioned machine learning techniques which typically train algorithms/networks with known ground truth labels, Reinforcement Learning (RL) offers an alternative way for artificial agents, software or robotic, to discover optimal ways of performing certain tasks. The intuition is that in some cases we won't be able to have a clear vision of what actions of an agent are "correct", but can only tell whether a task completed by the agent is desirable, or "rewardable". In other words, the agents won't be instructed with the sequence of actions to perform in order to complete the task, but rather if a particular sequence of actions was good enough or not. Reinforcement learning is designed particularly for those scenarios, where instead of testing every possible action sequences to find the appropriate one, it uses states as a mechanism to remember what happened during a sequence of actions, and iteratively propagates the rewards received at the end of the task (when a sequence of actions are completed) to the actions of the very sequence. Conceptually, RL is a family of machine learning methods that learns the decision making policies for agents while they interact with the environment and perform required tasks. According to the number of agents involved in the learning process, RL approaches can be categorised into two classes: the *single-agent* and the *multi-agent* frameworks.

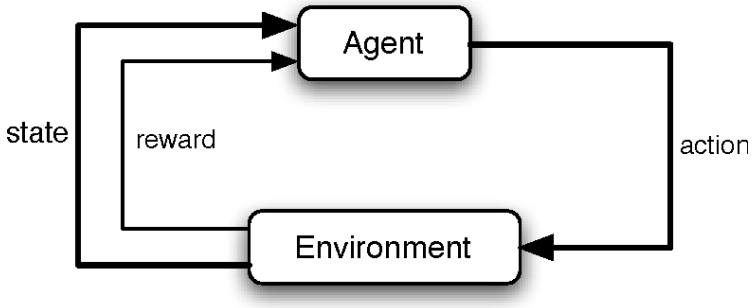


Figure 2.5: The general single-agent reinforcement learning framework.

In the following, we briefly introduce the general formulation of the two classes of RL approaches, and then discuss a recent popular RL paradigm, deep reinforcement learning, which is also adopted in this thesis, and its applications in the urban mobility sector.

2.3.1 Single-agent Reinforcement Learning Framework

As suggested by its name, the single-agent RL framework considers only one agent globally, which interacts with the environment by taking actions and then perceives the effects of those actions. The environment in return, updates the agent with a new state after actions are performed, and a reward/penalty measuring how the actions contributed to the specific goal. Therefore in this case, the objective of the agent is to maximise certain forms of the reward, e.g., sum of all rewards, and the RL algorithm aims to adjust the behaviours of the agent to achieve that. This can be modelled with the *Markov Decision Processes* (MDPs) [98], which is the basis for typical RL frameworks. In particular, a MDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ where:

- $\mathcal{S}$ is the set of states;
- $\mathcal{A}$ is the set of possible agent actions;
- $\mathcal{T}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is the state transition function, defined as a probability distribution over the states: $\mathcal{T}(s, a, s') = p\{s_{t-1} = s' | s_t = s, a_t = a\}$. s_t is the state at time t, a_t represents the action taken after observing state s_t and s_{t+1} indicates the new state at time $t + 1$;
- $\mathcal{R}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathcal{R}$ is the reward function, which determines the expected value of the reward given the current state s, action a and the resulting next

state s': $\mathcal{R}(s, a, s') = E\{r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'\}$. Here r_{t+1} is a specific reward at $t + 1$ given s_t and s_{t+1}, over which $\mathcal{R}$ computes the expectation.

In the above formulation, the state transitions $\mathcal{T}$ and reward function $\mathcal{R}$ are static, i.e., they won't change over time. Therefore, in this case the agent performs an actions a and collects responses from the environment, in the form of the subsequent state s' and resulting reward r. Then for RL, the task of deciding which action $a \in \mathcal{A}$ to choose is governed by a policy function, which is often denoted as π. Concretely, a policy $\pi = \{s_0, a_1, s_1, ..., a_{T-1}, s_T\}$ is a sequence of probability distributions, defining the probability that each action shall be chosen given the states. The key goal of RL is to find the optimal policy which will maximise the expected reward over time [99]. Broadly speaking, there are two classes of RL approaches to solve that: the *model-based* and *model-free* RL approaches.

Model-based RL. A model-based RL approach often learns the model of the system/environment explicitly, and then uses methods such as dynamic programming [100] to compute the optimal policy given the estimated model. Classical model-based RL includes value iteration and policy iteration, where the former tries to determine the optimal value function (expected reward for some state given the agent is following certain policy) that can lead to the optimal policy [99]. On the other hand, rather than finding the policy through value functions, the policy iteration approach manipulates the policy directly, which typically consists of two alternating steps: policy evaluation and policy improvement. The idea is that we firstly evaluate the state values corresponding to the current policy, and then improve the current policy in the greedy way with respect to the current value function. In practice, policy iteration could end up with fewer iterations than the value iteration approach, but it could also be slow in each iteration. Thus it depends on the specific application scenario to choose between the two methods.

Model-free RL. The above model-based RL approaches typically assume that the state transition $\mathcal{T}$ and reward function $\mathcal{R}$ are known, which in many cases is not possible, especially if we only want to learn the optimal policy while interacting with the environment. Therefore, the model-free RL approaches try to learn the state value function, and derive the optimal policy from this estimated function. The simplest algorithm in this context is the temporal difference (TD learning [99]). The idea is to use the immediate reward signal to approximate the state value function, which is essentially a kind of stochastic approximation. It shares similar policy evaluation step as the policy iteration approach, but the difference is that in TD we never improve the policy. There are two different types of learning approaches in this category, off-policy and on-policy, depending on if the policy being evaluated and

improved will be used to control the MDP or not. A classic off-policy method that has been widely used is Q-learning [101], which learns the optimal Q-values, instead of the state values, to find the optimal policy. The Q-values provide information on the future quality of actions, based on which the agent will select the behaviour policy in a probabilistic and greedy way, typically according to a parameter ϵ. On the other hand, the on-policy methods, such as Sarsa [102], operate according to the same policy which has being improved.

2.3.2 Multi-agent Reinforcement Learning Framework

The multi-agent RL framework is based on the same idea of its single-agent counterpart, but involves more than one agents interacting with the environment and each other. Compared to the single-agent RL, the challenge is that the actions from each agent will have some effect on the environment, and thus the outcome may depend very much on what the other agents do. Therefore, the single-agent based RL techniques cannot be directly applied here since they are designed to solve stationary environment. From each agent's point of view, in the multi-agent settings the environment is no longer stationary. Therefore, the multi-agent RL framework typically model the process with *stochastic games*, which can be viewed as an extension to the single-agent MDP in a sense as they deal with multiple agents in multiple states, involving compromises and cooperations. Formally, they can be defined as a tuple $(n, \mathcal{S}, \mathcal{A}_{1,...,n}, \mathcal{T}, \mathcal{R}_{1,...,n})$, where:

- n is the number of agents in the environment;
- $\mathcal{S}$ is the joint states;
- $\mathcal{A}_i$ is the set of possible actions of agent i and $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_n$ is the joint action space;
- $\mathcal{T}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is the transition function depending on the actions of all agents;
- $\mathcal{R}$: $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathcal{R}$ is the reward function representing the expected value of the next reward, which also depends on the action of all the agents.

In this multi-agent settings, for a given agent i, a policy π_i is still a collection of probability distributions over the available actions, one for each state. The collection of policies, one for each agent, is called the *jointly policy*. In the trivial case where there is only one agent, such a stochastic game can be reduce to a MDP, which

can be solved with the single-agent methods discussed above. On the other hand, two types of approaches are designed to solve the stochastic game: the *best-response learners* and *equilibrium learners* [99]. In particular, the former tries to learn the optimal policy with respect to the policies of the other agents, without consider equilibrium, while the latter aims to find the policies that are Nash equilibrium for the stochastic game [99].

2.3.3 Deep Reinforcement Learning

In practice, the state space $\mathcal{S}$ in many problems can be complex and of high dimensional, which often require manual engineering in traditional RL approaches. Deep reinforcement learning combines reinforcement learning and deep learning, which has shown significant advantages over the traditional approaches, e.g., it can process unstructured and large input data, such as raw images captured by cameras and sensor data recorded by robots, and decide what actions to perform given the objectives. Deep RL has been widely applied and enabled remarkable progresses in a number of research areas, from playing challenging games [103], to robotics [104] and computer vision [105].

Similar to the traditional RL approaches, deep RL can also be categorised into **model-free** and **model-based** methods. In model-based deep RL, a forward model of the environment can be built using a deep neural network, which can be trained by supervised learning. The actions of the agents can then be obtained from the learned model. The major benefit of model-based deep RL is that it can significantly reduce the sample complexity, while achieving the predictive power [106]. On the other hand, model-free deep RL approaches learn policies from interactions (i.e., samples) between the agent and the environment, without explicitly modelling of the forward model of the environment. A well known deep RL approach is the deep Q-network (DQN) [107], which uses a deep network to approximate the Q function in the traditional Q-learning. On the other hand, policy optimisation/gradient approaches, such as Proximal Policy Optimisation (PPO) [108], compute the policy directly with deep neural networks. The actor-critic methods, e.g., A2C [109], typically employ two deep neural networks, one critic and one actor, where the critic learns the value function (measuring how good the action taken is), while multiple instances of actors (controlling how the agent behaves) are trained in parallel.

2.3.4 Applications in Mobility Systems

Reinforcement learning, especially deep RL has also been used in the mobility context, due to their unprecedented performance in many domains. Existing work has shown that various challenging mobility problems that involve elements of actuation, such as traffic control [51], fleet management [13, 12] and rebalancing [40], can be modelled and addressed with the deep RL frameworks. Given their distributed nature, in most of the existing works, the mobility systems are modelled with the multi-agent RL framework, e.g., the vehicles/bikes in the mobility systems are considered to be independent agents, interacting with both the urban environment and the other agents. For instance, the work in [110] designs a spatio-temporal reinforcement learning approach to dynamically reposition bikes in bike-sharing systems. The idea is to use deep Q-Network (DQN) in the spatio-temporal domain to generate the optimal reposition actions. DQNs have also been used in fleet management related tasks such as order dispatching in taxi and ride sharing systems [12], where the work in [13] incorporates mean field approximation to improve efficiency. It has been shown that for mobility systems, or the general urban computing problems that involve agents interacting and actuating the environment, deep RL often demonstrates stronger performance than traditional approaches, particularly for high dimensional input and complex problem structures.

2.4 Discussion

As discussed in this chapter, the maturity of various mobility modalities and the remarkable advances in machine learning techniques for spatio-temporal data have significantly transformed the way that people develop, operate and optimise the shared mobility systems in urban settings. However as surveyed in this chapter, we also identified the following key limitations of the prior art that motivate this thesis:

- In the real-world, shared mobility systems are never static, but often change over time. This is overlooked by existing work, most of which only considers static systems, or a few static snapshots. The dynamics, e.g., expansion of the system infrastructure, have not been thoroughly studied in their analysis. In this thesis, we relax this oversimplified assumption, and assume shared mobility systems can dynamically update their states across the spatio-temporal domains. For all the tasks tackled in this thesis, from demand forecasting in Chapter 3, to infrastructure optimisation in Chapter 4 and fleet management in Chapter 5, we consider the shared mobility systems as *dynamic*, in the

sense that their components e.g., station networks and vehicles, can continuously evolve over space and time. We present the challenges associated with this dynamic settings in those tasks, and show that our approaches can cope well.

- Although various forms of spatio-temporal analysis have been applied in urban computing scenarios, deep learning techniques bespoke for shared mobility systems are still limited. In this thesis, we aim to design suitable modelling and analysis techniques that are specifically crafted for such systems, that can capture their unique properties in spatio-temporal domains, scale gracefully on large data volumes, and be resilient to real-world dynamics. Motivated by this, we present our dynamic multi-graph GCN approach in Chapter 3, which models the shared mobility systems with time-varying graphs and performs inference on them with tailored GCNs. Although designed for forecasting tasks, our approach can potentially be extended to other applications such as flow estimation and event detection.

- The infrastructure of shared mobility system plays an important role in their operations, and therefore optimisation of such infrastructure is vital. Existing works tackle this line of tasks has two major shortcomings. Firstly, most of existing solutions optimise the system in an once-for-all fashion, i.e., they compute the optimal plan for infrastructure deployment initially, and once the infrastructure is deployed, it stays unchanged as the systems are in operation. Secondly, existing approaches often use heuristic based optimisation in this process, which although simple to implement, could be easily stuck in local minima. We address such limitations in Chapter 4 by developing a deep neural search approach that is able to discover optimal deployment plans through space and time, guiding the mobility systems to dynamically adjust its infrastructure in order to achieve the desired goals.

- Rebalancing of the fleet is an important task for shared mobility systems. As mentioned above, most of the existing works assume that the mobility systems are static while actuating them, e.g., repositioning the fleets. However, this won't work well when the systems are dynamically changing in real-world scenarios, i.e., the decision space (where to reposition a vehicle) could have already changed during the operation. In addition, for mobility systems with specific constraints, e.g., those using electric vehicles, there is very limited work. Those systems, although sharing many common properties with standard ones, present unique challenges, e.g. electric vehicles typically have much

limited range with long charging time, and thus can't be repositioned to arbitrary locations nor at any time unless sufficiently charged. We tackle those limitations and challenges in Chapter 5, which explicitly models the dynamics and constraints of the shared mobility systems and tailors a deep reinforcement learning approach for this fleet management tasks.

We are now in a position to present the main contributions of this thesis in the following three chapters.

Chapter 3

Demand Forecasting

3.1 Introduction

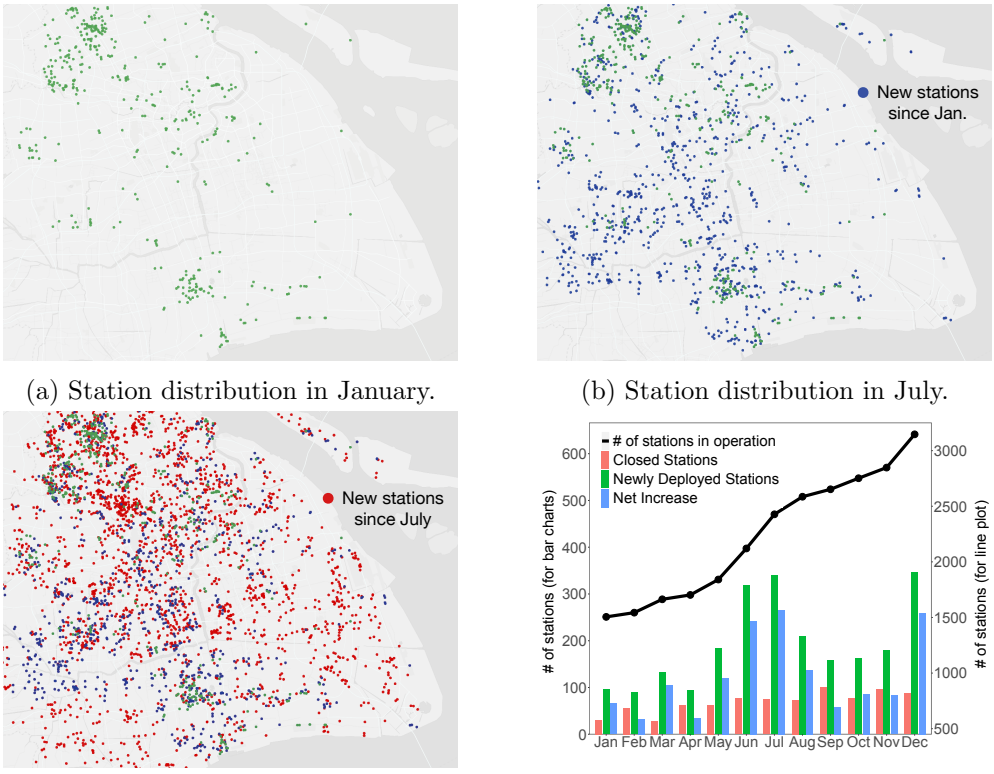
Cities around the globe struggle with congestion and poor air quality. Shared mobility systems are emerging within the cities as a new form of urban mobility forms, which have been recognised as an environmental friendly mobility option, reducing vehicles on the road while cutting out unnecessary CO_2 emissions. In particular, in the following chapters of this thesis, we consider a specific type of shared mobility systems, the station-based *shared e-mobility systems*, as the instantiation of our study. Those systems, as a new generation of mobility systems, have been increasingly popular across the globe, and taken significant market shares in the business, by offering full electric vehicle (EV) fleets with fast expanding infrastructures in major cities, e.g., Bluecity [3] in London, WeShare [4] in Berlin, and BlueSG [5] in Singapore. Traditional car sharing providers have also started to populate their EV fleets, e.g., ZipCar [32] seeks to provide over 9,000 full electric vehicles across London by 2025. According to a recent study [111], the global market of EV sharing services is poised for an even faster growth in the near future, due to the incentives and regulations put in place by governments across the world to encourage overall EV usages. Shared e-mobility systems will reshape the current urban transportation paradigm, providing a much more efficient, sustainable, and affordable mobility option to all citizens, independent of individual car ownership. Note that it is straightforward to apply the approaches presented in this thesis to the general shared mobility systems, i.e., by relaxing the constraints related to EVs such as range limits and charging models. Therefore without loss of generality, in the following three chapters we consider shared e-mobility systems in our context.

In this chapter, we study one of the key problems in shared e-mobility sys-

tems, demand forecasting, i.e., we would like to accurately predict the mobility demand of the users of the systems. As highlighted in previous chapters, a major problem of those systems is that during their fast expansion processes, dynamic predictions of user demand and implementing any expansion strategy becomes substantially more difficult. This is not only key for stakeholders to make informed decisions as to where and when to deploy new stations or close the poorly performing ones, but also of great importance to the effective operation of currently used stations: understanding the potential impact of proposed expansions to their demand can provide valuable insights on a number of vital tasks such as scheduling, pricing and rebalancing.

However, in the context of such fast expanding shared e-mobility systems, this demand prediction problem is not trivial. Most of the existing works on demand prediction [112, 113] assume the stations in the system are static and historical data is always available, or only predict demand after fixed (one or two) expansion stages where stations are only deployed in batches [114]. These assumptions often collapse in the real world. For instance, Fig. 3.1a-c visualise the expansion process of a major shared e-mobility platform in Shanghai during 2017. We see that in the beginning stations are only scattered within limited areas, while at the end of the year the entire city has been densely covered. As shown in Fig. 3.1d, within 12 months the total number of stations in operation has doubled (from roughly 1500 to more than 3000). In addition, we see that in each month there are continuously hundreds of stations being deployed or closed. In this case, predicting demand at those newly deployed or to be deployed stations is very challenging, since there is no sufficient historical data available as prior knowledge.

On the other hand, the new dynamics caused by the expansion process may have complex effects on the entire shared e-mobility system. For example, as shown in Fig. 3.2, deploying stations at various places may have completely different effects. For example, the new station (denoted as the red dot) in Fig. 3.2a "steals" the demand from one of its neighbors (station A) since its deployment in June (see the changes of their order numbers in Fig. 3.2b). We found from the data that the new station was deployed in a major shopping center, and therefore it likely attracted the users who originally preferred to rent/return cars from/to station A, which is just one block away. In contrast as shown in Fig. 3.2c and Fig. 3.2d, deploying a new station has increased the orders of its neighbor stations E, F and G collectively. In particular, we found from the data that a large portion of their increased orders have the new station as the destination. This means that after the new station is deployed, many users tend to rent EVs from E, F, and G because together with



(a) Station distribution in January.

(b) Station distribution in July.

(c) Station distribution in December.

(d) Stations statistics over 12 months.

Figure 3.1: The expansion process of a shared e-mobility system in Shanghai during the year. Images better viewed in colour.

the new station, they offer convenient short-range connections for them to get to the other side of the airport. In the presence of such dynamics, accurate demand prediction for the remaining stations becomes very challenging, due to the non-trivial system dynamics caused by the continuous expansion process.

To address those challenges, in this chapter we present a novel data-driven demand prediction approach, which models the expansion of shared e-mobility systems with time-varying graphs, and is able to forecast the accurate demand of stations along with the expansion process. Specifically, for each station that comes in operation, we employ a local temporal encoding module to capture the correlations within the historical data. The extracted features from all stations are then compiled by a dynamic spatial encoding module, which considers the spatial dependencies between them as multiple graphs, and fuses the station-level features with Graph Convolutional Neural Network (GCN). Based on the encoded information and future expansion plan (i.e., which stations to be deployed or closed), we consider a multi-scale predictor which forecasts station demand at different scales:

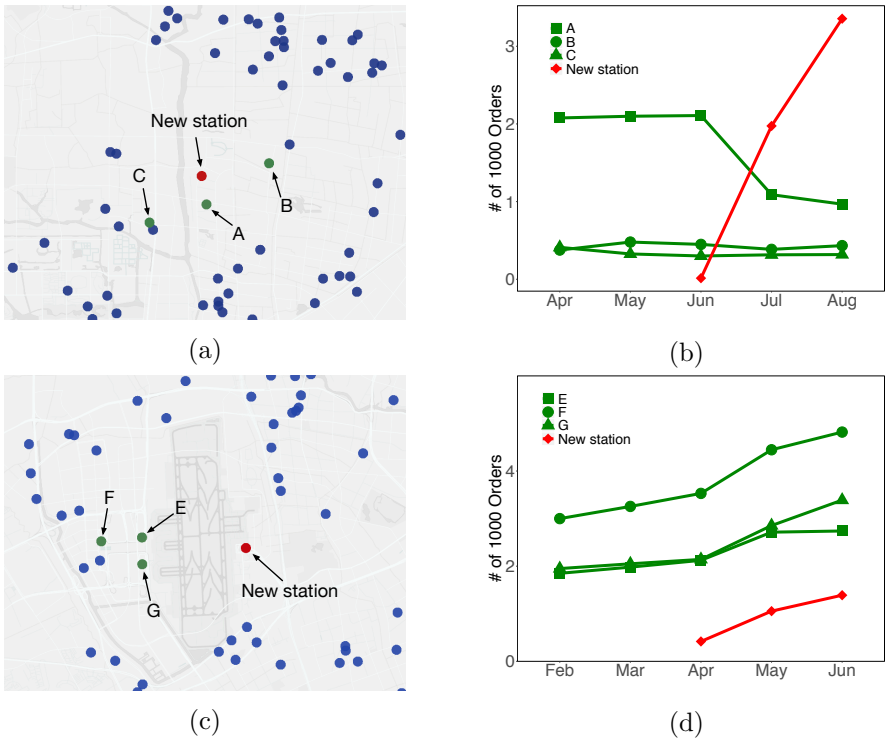


Figure 3.2: Different types of impact when deploying new stations to the current station network. (a)-(b) An example showing that a new station 'steals' the user demand from one of its neighbour stations. (c)-(d) An example showing that a new station 'boosts' the demand of its neighbour stations.

From instant demand in the immediate near future to the long term expected demand, for both stations to be deployed and the ones remaining. In summary, the technical contributions are as follows:

- To the best of our knowledge, this is the first work that investigates the demand forecasting problem in the context of fast expanding electric vehicle sharing systems. We conduct a comprehensive study with the operational data from a fast growing shared e-mobility system in the real world, and identify the needs and benefits of forecasting the accurate user demand as the system continuously expands, which have not been studied before.

- We propose a novel data-driven approach for demand prediction which is capable of modelling the complex dynamics caused by the fast system expansion. The key idea is to model the evolving station network of the shared e-mobility system as multiple time-varying graphs, which describe the different types of correlations between the station. With those graphs, we propose new encod-

ing approaches which perform the local temporal encoding and global spatial encoding processes in tandem, to jointly incorporate the historical knowledge at individual stations and the spatial dependencies between them.

- We design a new multi-scale predictor on top of the encoding processes, which is able to forecast the accurate user demand of both stations to be deployed and those already existing in the current system. In addition, our predictor can predict not only the expected future demand of the stations, but also their instant demand in subsequent timestamps, which allows us to better understand both short and long term impact of the system expansion.
- We evaluate the proposed demand prediction approach on both real and synthetic data collected from a major shared e-mobility platform in Shanghai for one year, including data from over 3000 stations and 16,000 electric vehicles in operation. Extensive experiments have shown that our approach significantly outperforms the state of the art, offering up to three-fold improvement in prediction accuracy and is robust to different levels of expansion dynamics.

The rest of this chapter is organised as follows. We first discuss additional related work in Section 3.2 and formulate the problem of demand forecasting for fast expanding system such as shared e-mobility systems in Section 3.3. Then we present the proposed data-driven demand prediction approach in Section 3.4, covering the proposed local temporal and dynamic spatial encoding techniques, as well as the design of a multi-scale predictor. Section 3.5 evaluates the performance of our approach with real-world shared e-mobility system data. We conclude the chapter in Section 3.6.

3.2 Related Work

In this section, we discuss additional related works that are particularly relevant to the contributions of this chapter. For general background we refer the readers to Chapter 2.

3.2.1 Demand Prediction for Shared Mobility Systems

Predicting user demand in shared mobility services (e.g., taxis and bike- or vehicle-sharing systems) has received considerable interests in various research communities. Most of the existing works take the historical usage (e.g., picking-up and returning records), geospatial data such as POIs, and other auxiliary information (e.g.,

weather) into account, and build prediction models that can forecast demand over certain periods or aggregated time slots. They also predict the demand at different spatial granularity, e.g., over the entire systems [115, 116], grids/regions [97], station clusters [117, 112, 118], or individual stations [34, 119, 120]. This work falls into the last category since we aim to predict station-level demand of shared e-mobility systems. However, our work is fundamentally different in that we assume the station network is not static, but dynamically evolving, i.e., stations can be deployed or closed at arbitrary times. This has not been investigated by the existing works, and in this case, state of the art station-level demand predictors (e.g., [113]) do not address our problem because they rely heavily on station historical data to make predictions, which are not available for those newly deployed stations.

3.2.2 Shared Mobility System Expansion

There is also a solid body of work focusing on modeling the expansion processes of shared mobility systems, e.g., planning for optimal new stations [49, 34, 121], or increasing the capacity of existing stations [10]. However, most of the existing works assume that historical information on demand of the stations (renting and returning) are either known, or can be estimated from other data sources such as taxi records, which is fundamentally different from our work. On the other hand, the work in [114] proposes a functional zone based hierarchical demand predictor for shared bike systems, which can estimate the average demand at newly deployed stations across different expansion stages. In general, our work in this chapter shares similar assumptions with [114], yet differs substantially: 1) instead of fixed stages, we can predict demand while the entire station network is dynamically expanding; 2) we are able to estimate both the instant and expected demand of new or existing stations, while [114] can only predict aggregated demand patterns; and finally 3) we do not require historical mobility data in the newly expanded areas, like the taxi trip records required in [114].

3.2.3 Graph-based Learning for Urban Prediction

Our work shares similar challenges and settings with the other urban analysis problems, such as crowd prediction [122, 123], traffic analysis [124], anomalies detection [125, 126]. Due to their non-Euclidean nature, recently many real-world problems such as demand/traffic/air quality forecasting that require spatio-temporal analysis have been tackled with the emerging graph-based deep learning techniques [96, 52]. In particular, existing works often employ the graph convolutional neural net-

work [92] to capture the spatial correlations, where temporal dependencies are typically modelled with recurrent neural networks. We consider a novel dynamic GCN approach that can handle the time-varying graph of the growing EV station network), while most of the recent work, e.g., those on crowd prediction [123, 122] assume the space (modeled as graphs where different regions are nodes) is stationary. Essentially, the graphs considered in the crowd prediction work can be viewed as static containers of the crowd, as they represent regions of the space. Thus their approaches cannot be applied to our demand prediction problem where the graphs themselves are variables evolving over time. In addition, the work in [96] models the traffic flow as a diffusion process on directed graphs for traffic forecasting, while [52] and [97] propose frameworks that use multi-graph convolutional neural networks (CNNs) to predict demand for taxi and ride-hailing services. Another work in [11] uses an encoder-decoder structure on top of multi-graph CNNs to estimate flow between stations in bike sharing systems, which bears a close resemblance to this work. However, unlike [11] who only output demand at the immediate next timestamp, our work considers a sequence to sequence model with attention mechanism to perform multi-step forecasting towards future demand. None of the above approaches can work on new stations where historical data is not available, i.e., they do not address continuous system expansion.

3.3 Problem Formulation

In this section, we first introduce some key concepts used throughout this chapter, then we formulate the problem of demand prediction for expanding shared e-mobility systems and provide an overview of the proposed framework. Note that in the following chapters we will also provide additional details relevant to the specific problems tackled when necessary.

3.3.1 Preliminaries

Electric Vehicle (EV) Stations. Let s_i be a station in the shared e-mobility system. In this chapter, we assume s_i can be represented as a tuple $(\boldsymbol{x}_i, m_i)$, where $\boldsymbol{x}_i$ are the geographic coordinates (e.g. latitude and longitude) of station s_i, and m_i is the number of charging docks within s_i. We also assume that for a given s_i, we can extract a number of geospatial features based on its location $\boldsymbol{x}_i$, such as nearby Points of Interest (POI) or the distribution of road networks within a certain radius.
Instant Station Demand. We define the instant demand of a station s_i at timestamp t as the rent/return frequency of s_i, denoted as $d_i(t)$. In this chapter the

granularity of timestamp t is days, i.e., we focus on daily station demand, but the proposed approach can be extended to adopt other time granularity levels.

Expected Station Demand. For a station s_i, the expected demand $\bar{d}_i$ over a period $[t_s, t_e]$ can be defined as the mean $\bar{d}_i(t_s, t_e) = |t_e - t_s|^{-1} \sum_{t=t_s}^{t_e} d_i(t)$. We consider the expected demand from the current time t towards the future, and aggregate it according to some index, e.g., days of the week. Without loss of generality, in the following text we denote the future expected demand of station s_i as a vector $\bar{\boldsymbol{d}}_i = [\bar{d}_i^{\text{Mo}}, \bar{d}_i^{\text{Tu}}, ..., \bar{d}_i^{\text{Su}}]$, $\bar{\boldsymbol{d}}_i \in \mathbb{R}^7$ for different days of the week.

Station Network. We model the stations of the shared e-mobility system as a graph $G = (S, A)$, where the nodes $s_i \in S$ are stations as defined above. An edge $a_{ij} \in A$ may encode a certain type of correlation between two stations s_i and s_j, e.g., the spatial distance between them, or similarity between their POI/road network features. Section 3.4.2 will discuss how we construct multiple graphs to capture such inter-station relationships in more detail.

Station Network Dynamics. Unlike existing work, in this thesis we assume the station network is *evolving* over time, i.e., $G = (S, A)$ is a time-varying graph. More specifically, let $G_{t-1} = (S_{t-1}, A_{t-1})$ represents the station network at time $t-1$. Without loss of generality, we assume that at time $t-1$, there is an expansion plan to be implemented at time t, which shall expand the current station network from G_{t-1} to the *planned network* G_t^{P}. Let's assume during this a set of new stations S^+ will be deployed, while existing stations S^- will be closed. If the expansion plan goes through, then at time t the actual station network $G_t = (S_t, A_t)$ becomes the planned G_t^{P}, where

$$S_t = (S_{t-1} - S^-) \cup S^+ \tag{3.1a}$$

$$A_t = (A_{t-1} - \{a_{ij} | s_i \in S^- \text{ or } s_j \in S^-\}) \cup \{a_{ij} | s_i \in S^+ \text{ or } s_j \in S^+\} \tag{3.1b}$$

3.3.2 The Demand Prediction Problem

Suppose that at time t, we have the topology $G_1, ...G_t$ and demand $D_1, ..., D_t$ of the station network, where $D_t = \{d_i(t) | s_i \in G_t\}$. Let G_{t+1}^{P} be the planned station network at the future timestamp $t+1$. The demand prediction problem addressed in this chapter is that given the historical data, for an arbitrary station in the planned network $s_i \in G_{t+1}^{\text{P}}$ (deployed or not yet deployed) we aim to estimate both its expected future demand $\hat{\bar{\boldsymbol{d}}}_i$ and the subsequent k instant demand $[\hat{d}_i(t+1), \hat{d}_i(t+2), ..., \hat{d}_i(t+k)]$, which minimise the mean square errors with respect to the ground

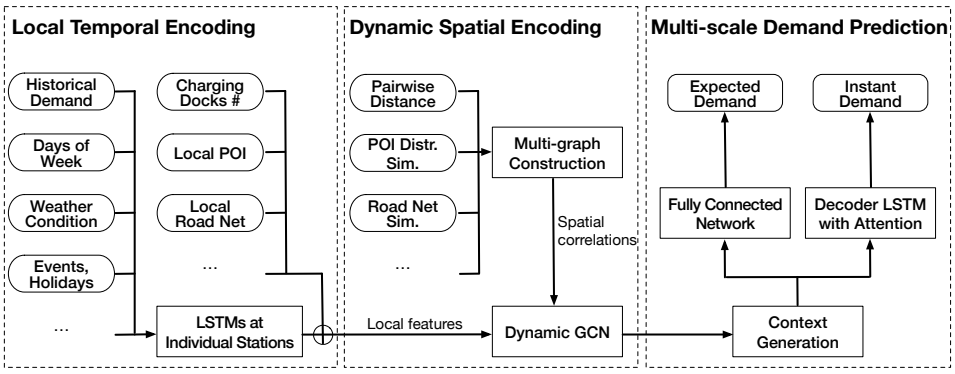


Figure 3.3: Overview of the proposed data-driven demand prediction framework.

truth $\bar{\boldsymbol{d}}_i$ and d_i:

$$\delta_{\bar{\boldsymbol{d}}_i} = |\bar{\boldsymbol{d}}_i|^{-1} \|\hat{\bar{\boldsymbol{d}}}_i - \bar{\boldsymbol{d}}_i\|^2 \tag{3.2a}$$

$$\delta_{d_i} = k^{-1} \sum_{\tau=t+1}^{t+k} \|\hat{d}_i(\tau) - d_i(\tau)\|^2 \tag{3.2b}$$

In practice, the expected demand $\hat{\bar{\boldsymbol{d}}}_i$ can be viewed as a metric for the *long-term performance* of stations s_i, e.g., if s_i is a station to be deployed, $\hat{\bar{\boldsymbol{d}}}_i$ quantifies the average level of demand it may be able to attract. On the other hand, the sequence of instant demand $[\hat{d}_i(t+1), \hat{d}_i(t+2), ..., \hat{d}_i(t+k)]$ describes the *immediate trend* of station demand under the impact of the expansion plan, which can help to optimise key future operation strategies such as marketing and resource allocation.

3.3.3 Framework Overview

Fig. 3.3 shows the overview of the proposed data-driven demand prediction framework, which consists of three major components:

Local Temporal Encoding. During the life cycle of a station s_i (from being deployed to shut down), its demand can be viewed as a time series, where the current demand $d_i(t)$ should correlate with the local historical demand $d_i(t-1), ..., d_i(1)$. In addition, there may exist other temporal factors that can influence the demand of individual stations, such as weather conditions, air pollution levels, days of the week and public holidays etc. To model such temporal dependencies, we assign a Long Short-Term Memory (LSTM) network at each individual station when being deployed, and use them to encode local temporal information at station level.

Dynamic Spatial Encoding. Intuitively, the demand of a station s_i can be affected also by the other stations in the network. To capture the spatial correlations,

at each time t we construct multiple graphs to encode different spatial relationships between the stations, e.g., inter-station distances, POI similarity, and road network metrics. Then we use a Graph Convolutional Neural Network (GCN) to fuse those graphs and encode the previously computed local features of individual stations. In particular, as the station network is evolving over time, we develop a dynamic GCN (DGCN) which is able to process such time-varying graphs.

Multi-scale Demand Prediction. Based on the results of the above temporal and spatial encoding, we aim to predict both the expected demand and subsequent instant demand of stations after the planned expansion. To achieve that, we design a multi-scale prediction network, which firstly compiles the previously learned features into a context vector. For expected demand, it uses a fully connected branch to perform the prediction, while on the other hand, it considers a decoder LSTM network with attention mechanism to forecast instant demand at multiple future timestamps.

We are now in a position to elaborate the proposed data-driven demand prediction approach in more detail.

3.4 Data-driven Demand Prediction

In this section, we present the proposed data-driven demand prediction approach, where we first introduce the local temporal and dynamic encoding techniques in Sections 3.4.1 and Section 3.4.2 respectively, and then in Section 3.4.3, we describe the design of the multi-scale predictor.

3.4.1 Local Temporal Encoding

Like in many other shared mobility services, we observe that the demand of stations in the shared e-mobility system exhibits strong temporal correlations, as shown later in Fig. 3.6b. For instance, although it fluctuates largely over time, the demand at an individual station approximates certain periodical patterns at different days across the week. In that sense, exploiting such knowledge can help significantly in estimating the future demand of the existing stations, which will have a positive knock-on effect when predicting demand for the new stations during expansion. However, those demand patterns are typically influenced by multiple factors such as weather, air quality and events, and individual stations may react to those factors differently. Therefore, it is often not optimal to only incorporate the temporal information globally for the station network, but instead we model such microdynamics at the station level.

Concretely, when a station s_i is deployed, we instantiate a LSTM network which keeps processing its demand records and the additional temporal information available, e.g., weather, days of the week and public holiday/events. In our implementation, we encode such temporal information as feature vectors, e.g., the weather data can be discretised and represented as one hot vector, and all the vectors are concatenated as the input to the LSTMs. To avoid over-fitting, we train the LSTMs with shared weights across stations. At time t, the LSTMs encode the station's historical demand $d_i(t), d_i(t-1), ...$ as well as the auxiliary information into a temporal feature vector $\mathbf{f}_i(t)$. Moreover, in this work we also condition $\mathbf{f}_i(t)$ with a static station feature vector $\mathbf{c}_i$, which describes key attributes of the station s_i such as its number of available charging docks m_i, nearby POIs and environmental characteristics, etc. We encode the static feature $\mathbf{c}_i$ in a similar way with the temporal information, e.g., the POI data can be represented as vectors where each element indicates the number of a particular type of POIs that are close to the station. Therefore, $\mathbf{f}_i(t)$ and $\mathbf{c}_i$ carry important local information about individual station since it started operating, which are concatenated and passed on as the input for the later spatial encoding. Fig. 3.4 shows the workflow of the proposed approach, where at each timestamp we maintain a collection of local LSTMs to encode information of individual stations.

3.4.2 Dynamic Spatial Encoding

Constructing Multiple Graphs

As discussed in Section 3.3.1, at a given time t we represent the station network as a graph $G_t = (S_t, A_t)$, where S_t are the set of current stations and A_t is the adjacent matrix describing the pairwise correlations between them. In practice there are often more than one types of correlations, which cannot be effectively captured by a single graph. Therefore in this work we construct multiple graphs to encode the complex inter-station relationships [97] particularly the *distance graph*, the *functional similarity graph*, and the *road accessibility graph* (see Fig. 3.4).

Distance. In most cases, we observe that the demand of stations close to each other are highly correlated, e.g., they may be deployed around the same shopping centre, and thus tend to be used interchangeably. We capture such correlations with a distance graph A^{D}, whose elements are the reciprocal of station distance:

$$a_{ij}^{\mathrm{D}} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^{-1} \tag{3.3}$$

where $\boldsymbol{x}_i$,$\boldsymbol{x}_j$ are the station coordinates, and $\|\cdot\|_2$ is the Euclidean distance. We

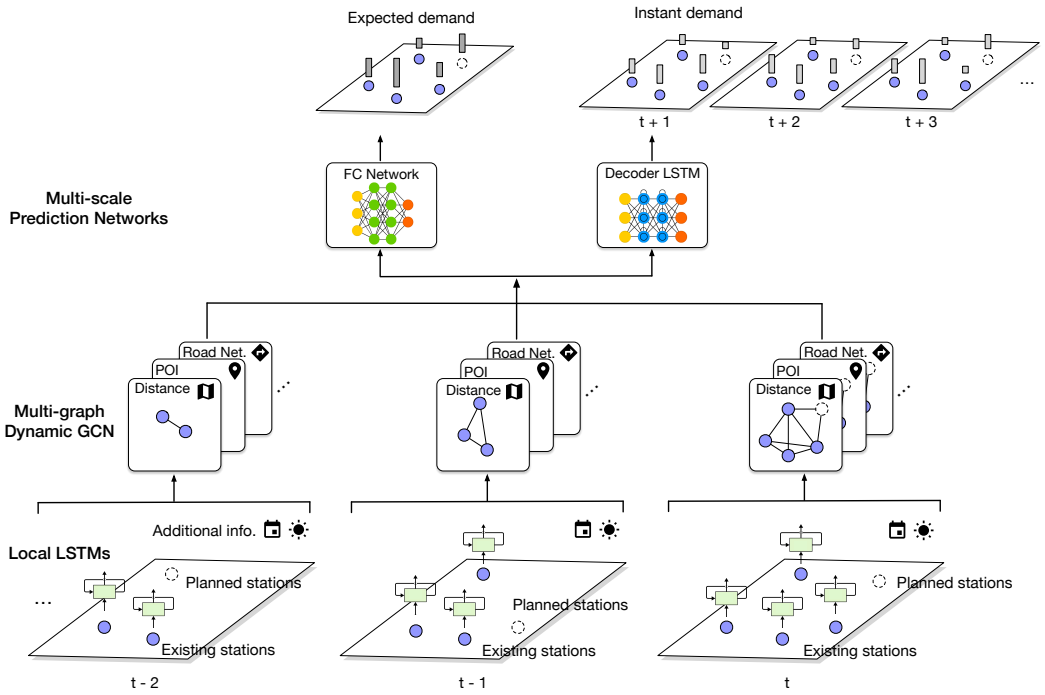


Figure 3.4: The workflow of the proposed data-driven demand prediction approach. The local temporal encoding process produces the temporal and static features $\mathbf{f}_i(t)$ and $\mathbf{c}_i$, which are concatenated as the input to the spatial encoding process with the multi-graph dynamic GCN. The output of the GCN $\boldsymbol{H}_t$ encodes both the spatial and temporal features across different stations, and are fed into the multi-scale prediction networks.

also set diag (A^{D}) to 1 to include self loops in the graph.

Functional Similarity. Intuitively, stations deployed in areas with similar functionalities should share comparable demand patterns. For instance, stations close to university campuses typically have significantly higher demand during weekends. We characterise the functionalities of stations by considering the distributions of their surrounding POIs. Suppose we have P different categories of POIs in total, and let $\boldsymbol{p}_i$ be the distribution of the P types of POIs within a certain radius of station s_i. The functional similarity graph A^{F} is then defined as:

$$a_{ij}^{\mathrm{F}} = \mathrm{sim}\left(\boldsymbol{p}_i, \boldsymbol{p}_j\right) \tag{3.4}$$

where $\mathrm{sim}\left(,\right) \in [0, 1]$ is a similarity measure which quantifies the distance between feature vectors. In our experiments, we use the soft cosine function.

Road Accessibility. Another factor that affects station demand is the accessibility

to road networks. Intuitively, stations close to major ring roads, or within areas that have densely connected streets would have higher demand. To model this, we consider the drivable streets in the vicinity of a station s_i as a local road network, containing different types of road segments and their junctions. We extract a feature vector $\boldsymbol{r}_i$ from the local road network, which encodes information such as the road segments density, average junction degree and mean centrality. Given those features, the road accessibility graph can be defined with a similarity function $\text{sim}(,)$:

$$a_{ij}^{\text{R}} = \text{sim}(\boldsymbol{r}_i, \boldsymbol{r}_j) \tag{3.5}$$

where we also use soft cosine as the similarity function.

Dynamic Multi-graph Convolution

At time $t-1$, given the constructed graphs $\boldsymbol{A}_{t-1} = \{A_{t-1}^D, A_{t-1}^F, A_{t-1}^R\}$ which describe the inter-station relationships, we propose a dynamic multi-graph GCN (DGCN) to fuse such spatial knowledge with local features $\mathbf{f}_i(t-1)$ and $\mathbf{c}_i$ computed by the station-level temporal encoding. In the proposed DGCN, we perform multi-graph convolution as follows:

$$\boldsymbol{H}_{t-1}^{(l)} = \sigma\Bigg(\sum_{A_{t-1} \in \boldsymbol{A}_{t-1}} f(A_{t-1}) \boldsymbol{H}_{t-1}^{(l-1)} \boldsymbol{W}_{t-1}^{(l-1)}\Bigg) \tag{3.6}$$

where $\boldsymbol{H}_{t-1}^{l-1}$ and $\boldsymbol{H}_{t-1}^{l}$ are the hidden features of layers $l-1$ and l respectively, while $\boldsymbol{W}_{t-1}^{l-1} \in \mathbb{R}^{U_{l-1} \times U_l}$ is the feature transformation matrix learned through end-to-end training. In particular, the input $\boldsymbol{H}_{t-1}^{(0)}$ is the collection of local features computed at individual stations. $f(A_{t-1})$ is a function on graphs A_{t-1}, e.g., the symmetric normalized Laplacian [127] or k-order polynomial function of Laplacian [97], and σ is a non-linear activation function such as ReLU.

As discussed before, in our case the station network evolves over time, i.e., new/existing stations can be opened or closed at any time. For simplicity, suppose at t there is only one new station s^N has been deployed. To capture this event, we recalculate the inter-station graphs $\boldsymbol{A}_{t-1}$ by appending new rows and columns to them, where the new graphs $\boldsymbol{A}_t$ now contain pairwise correlations between the new s^N and each existing stations. Note that the DGCN input also changes, i.e., now $\boldsymbol{H}_t^{(0)}$ has an extra feature for this newly deployed station s^N, computed by the local encoding process.

On the other hand, let s_j be the station that has been closed at time t. In our implementation, instead of removing elements from the graphs, we simply

apply a mask of zeros to the corresponding rows and columns of A_t, and set the j-th row of the input $\boldsymbol{H}_t^{(0)}$ to zeros since there won't be local features generated from s_j anymore. The intuition is that in our graph representation, $a_{:,j} = 0$ means station s_j has no correlation with any other station at all, and thus won't be able to propagate information in the graph convolution. Therefore in our case, at different timestamps the dimension of the input to our GCN can be different, i.e., the dimensions of the graphs $\boldsymbol{A}_t$ and input features $\boldsymbol{H}_t^{(0)}$ are varying. However this won't affect the learning process, since the learnable parameters $\boldsymbol{W}_t^l$ at each layer l have fixed dimensions. In addition, note that although $f(A_t)$ produces filters with the same size of the feature $\boldsymbol{H}_t^{(l)}$ at each layer l, Eq. (3.6) can still be viewed as a local convolution given the graphs $\boldsymbol{A}_t$. The reason is that by definition many elements in $\boldsymbol{A}_t$ are near zero (e.g., in the distance graph A_t^{D}), i.e., for a given station, it will be only affected by the features of stations with sufficiently high correlations with it (having large non-zero elements in $\boldsymbol{A}_t$). Conceptually, the dynamic GCN (DGCN) operates on snapshots of the inter-station graphs which are constructed on-the-fly, and fuses the local temporal features at individual stations with the spatial dependencies encoded in those graphs.

3.4.3 Multi-scale Demand Prediction

As discussed in Section 3.3.2, the demand prediction problem addressed in this chapter is to forecast the future demand of arbitrary stations in the shared e-mobility system under the planned expansion, given the historical data and previous dynamics of the station network. We have shown in the previous sections how we use local LSTMs and dynamic GCN (DGCN) to encode the spatial-temporal dynamics of the system, and in this section we explain how to make predictions of the user demand at multiple scales based on the knowledge extracted from the encoding processes. Fig. 3.5 shows the architecture of the proposed multi-scale demand prediction network.

Predicting Expected Demand

Let G_t be the current station network at time t. Without loss of generality, we assume that at the next timestamp we plan to deploy a candidate new station s^N, while will close an existing station s_j. Therefore, the goal is to predict the future demand of each individual station in this planned station network G_{t+1}^{P}. To achieve that, for each station in G_{t+1}^{P}, we run the LSTMs in the local temporal encoding process (Section 3.4.1) to generate an additional feature for time $t+1$, and create the

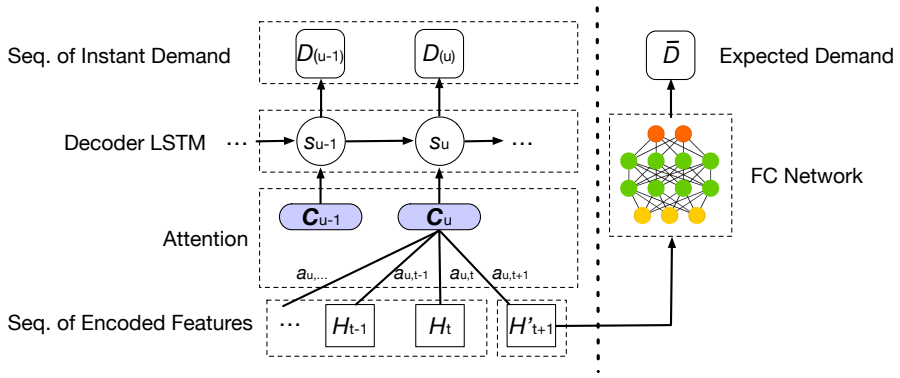


Figure 3.5: The proposed multi-scale demand predictor. Left: Decoder LSTM with attention mechanism for instant demand prediction. Here the decoder runs from timestamp $u = [t+1, ..., t+k]$. Right: Fully connected network for expected demand prediction. Note that $\boldsymbol{H}'_{t+1}$ is computed using the planned station network G^{P}_{t+1}, while $\boldsymbol{H}_t$, $\boldsymbol{H}_{t-1}$,... are derived from the actual data until time t.

new input feature $\boldsymbol{H}^{(0)'}_{t+1}$ for the GCN. Note that at this moment there is no historical data for the planned new station s^N since it is not deployed yet, and therefore here we only include its static features $\mathbf{c}_{s^N}$ while keeping its temporal features $\mathbf{f}_{s^N}(t+1)$ as zeros. We also mask the row corresponding to station s_j with zeros in $\boldsymbol{H}^{(0)'}_{t+1}$, to mute features from s_j which will be removed at $t+1$. Then we process the planned station network G^{P}_{t+1} by applying the same update to the inter-station graphs as discussed in Section 3.4.2, i.e., adding and masking the rows and columns corresponding to s^N and s_j. The generated feature $\boldsymbol{H}^{(0)'}_{t+1}$ is then passed through the multi-graph DGCN, producing an output $\boldsymbol{H}'_{t+1}$. We consider this $\boldsymbol{H}'_{t+1}$ as the *context* for prediction, because it not only encodes the current information about the new candidate station s^N and the spatial dependencies between stations, but is also relevant to the available historical information, since the underlying temporal encoding process uses LSTMs to preserve the temporal correlations.

In this work, we consider the expected demand of station s_i over different days of the week, indicating the mean demand that the station can attract in the future at each week day, i.e., $\bar{\boldsymbol{d}}_i = [\bar{d}^{\mathrm{Mo}}_i, \bar{d}^{\mathrm{Tu}}_i, ..., \bar{d}^{\mathrm{Su}}_i]$. To predict $\bar{\boldsymbol{d}}_i$, we plug in a fully connected network to the context vector $\boldsymbol{H}'_{t+1}$, which is trained to output the future expected demand (7 values indicating demand on different week days) for each station in the network G^{P}_{t+1}. For the station s^N, the predicted expected demand of itself and nearby stations indicate the potential benefits of deploying s^N to the current station network. In Section 3.5.3 we will show that in real-world experiments our approach significantly outperforms the existing techniques in prediction accuracy.

Predicting Instant Demand

We also predict the future instant demand of stations in the planned network G^{P}_{t+1} over a certain time window. This is also of great importance in practice, especially for the planned new station s^N, since it forecasts the immediate impact and future trends of the station network once s^N is in operation. However it is more challenging than predicting the expected demand, because essentially for each station we need to predict a sequence of concrete demand instead of the aggregated values.

To address that, we design a decoder LSTM network with attention architecture, which takes a sequence of previous features computed by the dynamic multi-graph GCN as input, and estimates the future k instant demand. In this case, conceptually the prediction framework becomes an encoder-decoder structure, where the processes of local temporal encoding and dynamic spatial encoding serve together as the encoder. Let $[\boldsymbol{H}_{t-n}, ..., \boldsymbol{H}_t, \boldsymbol{H}'_{t+1}]$ be the sequence of $n+1$ previous features generated by our DGCN. Unlike in the previous case where we only consider the last output feature $\boldsymbol{H}'_{t+1}$ as the context for prediction, here for each timestamp u in the prediction window of length k, i.e., $u = [t+1, ..., t+k]$, we construct the context vectors by fusing the feature sequence with attention mechanism:

$$\mathbf{Ctx}_u = \sum_{v=t-n}^{t+1} \alpha_{uv} \boldsymbol{H}_v \tag{3.7}$$

where α_{uv} are the attention weights determining the contribution of a feature $\boldsymbol{H}_v$ ($v \in [t-n, t+1]$) in predicting the demand at time u. Those weights α_{uv} are trained through back propagation in the end-to-end optimization. Then the decoder LSTM consumes the context vectors and predicts the k subsequent future demand. We found in our experiments that the attention mechanism is very helpful, since the station demand patterns tend to have strong periodic components, e.g., demand on this Monday is highly correlated with previous Mondays, and a single context vector is too compressed to encode such correlation. In our implementation we typically set $n = k$ or $n = 2k$ to better capture such periodical pattern in the station demand.

3.5 Evaluation

In this section, we evaluate the performance of the proposed data-driven demand prediction approach on data from a shared e-mobility system in Shanghai, China. We first describe the datasets, baseline approaches and implementation details of our experiments (Sections 3.5.1 and 3.5.2), and then discuss the experimental results

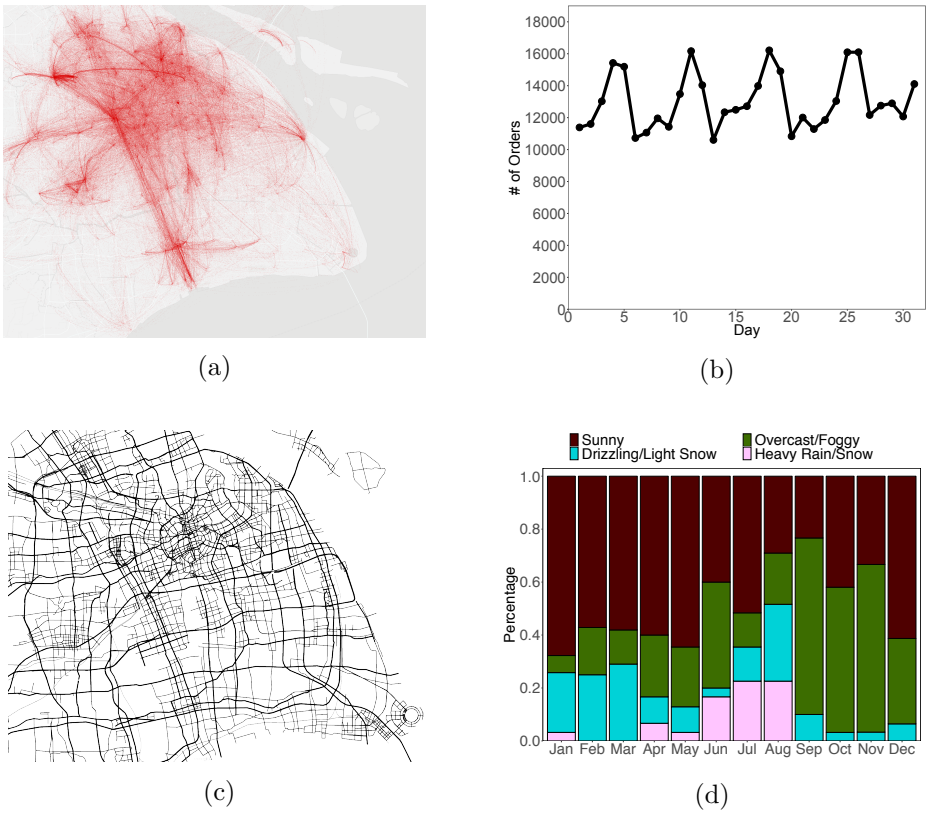


Figure 3.6: Visualisation of data used in the experiments. (a) Spatial distribution of orders (only showing the most frequent orders over the year). (b) Number of orders in one month. (c) Road network in Shanghai in a graph format. (d) Weather distribution of Shanghai in each month of the year.

in Section 3.5.3.

3.5.1 Datasets

Electric Vehicle (EV) Sharing Data. Our EV data is collected from real-world operational records of a shared e-mobility platform for one year (January to December 2017), containing two sets of data: i) the renting/returning orders at each stations, where an example of the order transaction are illustrated in Table. 3.1 and ii)the detailed expansion process of the station network (i.e., when and where a station was deployed/closed). In particular, there were 1705 stations and 4725 electric vehicles at the beginning of 2017, while as of December 2017 there were 3127 stations with a fleet of 16148 vehicles in operation. In total, the raw data contains 6,843,737 records, which were generated by approximately 0.36 million active users. Fig. 3.6a visualises the spatial distribution of the orders (represented

as lines between pick up and return stations) in a month. Fig. 3.6b shows the numbers of orders at different days-of-month, which exhibit clear periodic patterns with peaks on weekends.

Table 3.1: An example of the order transactions of the shared e-mobility platform.

Order_ID	20171108151609
User_ID	00068594305670
Pickup_Station	PUDONG_Airport
Pickup_Time	2017-08-09-11-54-07
Return_Station	HONGQIAO_Station
Return_Time	2017-08-09-13-08-51
Vehicle_ID	nfiehtiuwoqr
Pickup_mileage(remaining)	200
Return_mileage(remaining)	50

Synthetic System Expansion Data. In addition to the actual EV data, we also synthesise additional datasets for training purposes with different patterns of system expansion. The rationale is that the real expansion data only represents one sample (run) of system expansion, and it is not sufficient for our models to learn how to react to the general expansion process. Specifically, given the real EV data (both orders and expansion data), at each timestamp t we randomly pick a subset of existing stations according to a probability p, and treat those stations as newly deployed, i.e., assuming they don't have any previous order data. Note that here we select the stations randomly instead of following certain rules in order to make learning more generalisable. We vary the probability p from 0 to 1, generating multiple synthetic datasets ($p = 0.1$, ..., $p = 1$) with different expansion dynamics. Intuitively, the case where $p = 0$ is the real EV data without any extra injected expansion dynamics, while $p = 1$ is the extreme case where all stations at every timestamp are supposed to be newly deployed. Note that we only use the synthetic data in training, and for testing we always use the real data. As shown later in Section 3.5.3, the synthetic datasets effectively augment the real EV data, which help the proposed demand prediction approach to generalise better.

POI Data. We also collect Point Of Interest (POI) data from an online map service provider [128] in China. In total we have extracted 4,126,844 POI entries in Shanghai, each of which consists of a GPS coordinate and a category label. The label indicates the particular type and function of the POI, e.g., hospitals, subway stations, schools etc. In our experiments, for each station we only consider the POIs within 1km radius. Table. 3.2 shows the statistics of some POI categories. In our implementation we use one hot vector to represent the POI features of the stations,

Table 3.2: Statistics of some POI categories in our data.

POI Type	Number	POI Type	Number
Hospitals	4745	Banks	2988
Tourist attractions	2696	Companies	89,747
Gov. organizations	16,425	Higher education	6922
Airport services	126	Residences	51,089
Subway stations	1,729	Hotels	18,234
Bus stations	41,475	...	...

i.e., the elements encode the numbers of particular types of POIs which are close to the station.

Road Network Data. We extract road network data in Shanghai using OSMnx [129] from OpenStreetMap [130], which is formatted as a graph (visualised in Fig. 3.6c). Similar with the POIs, we consider the subgraphs within 1km radius of the stations, and compile key statistics such as mean degree, length of road segments etc. into the feature vectors. In our data, on average a subgraph contains road segments of length 13.85km and approximately 39 junctions, with a mean degree of 4.28.

Meteorology Data. Finally, we collect the historical daily weather data in Shanghai for the year 2017 from a publicly available source [131]. Each record describes weather conditions of the day, which falls into four different categories: *sunny*, *overcast/foggy*, *drizzling/light snow* and *heavy rain/snow*. Then natually we encode the data using one hot vector as the weather features. Fig. 3.6d shows the distribution of weather conditions in Shanghai over the 12 months.

3.5.2 Experimental Setup

We evaluate two variants of the proposed data-driven demand prediction approach respectively: 1) **D^{3}P-Exp**, which predicts the future *expected demand* of stations; and 2) **D^{3}P-Seq**, which forecasts the *instant demand* of stations in a subsequent time window. Both of the two variants share the same local temporal and dynamic spatial encoding processes, but they implement the two different branches in our multi-scale demand predictor and forecast future demand at different scales (as discussed in Section 3.4.3).

Competing Approaches. In particular, for predicting the expected demand, we compare our **D^{3}P-Exp** approach with the following baselines:

- **KNN** [132], which uses a linear regressor to predict the expected demand of existing stations. For the planned stations, it estimates their demand with

standard KNN, based on the similarity of features (e.g., POIs) between them and the existing stations.

- **Random Forest (RF)** [133], which shares the similar idea as KNN, but trains a random forest as the predictor.
- **Functional Zone (FZ)**, which implements the state of the art demand prediction approach for system expansion in [114]. Note that we don't have taxi records in our data, but instead we directly feed the ground truth check-in/out to favour this approach.

For **D^{3}P-Seq** which computes the instant demand, we consider three competing algorithms:

- **ARIMA + KNN**, which uses Auto-Regressive Integrated Moving Average (ARIMA) [134] to forecast multi-step demand at existing stations, and then uses KNN to estimate demand at new station based on station features such as POIs.
- **LSTM + KNN**, which is similar with A-KNN, but trains LSTM networks for temporal modelling.
- **Multi-graph convolutional network (MGCN)**, which implements a similar framework as the state of the art in [11], whose implementation is not publicly available. More importantly, the original framework in [11] is not able to work with time-varying graphs. Therefore, to perform fair comparison, here we use our dynamic multi-graph GCN implementations that can handle new/closed stations, and consider the same data sources as in our approach.

Evaluation Metrics. For all approaches, we adopt the *Root Mean Squared Error* (RMSE) and the *Error Rate* (ER) as the performance metric:

$$\begin{aligned} RMSE &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{z}_i - z_i)^2} \\ ER &= \frac{\sum_{i=1}^{N}|\hat{z}_i - z_i|}{\sum_{i=1}^{N} z_i} \end{aligned} \tag{3.8}$$

where $\hat{z}_i$ and z_i are the predicted and ground truth values respectively.

Implementation Details. We implement the deep neural networks in the proposed approach with TensorFlow [135] 1.10.0, and use the Adam optimiser [136]

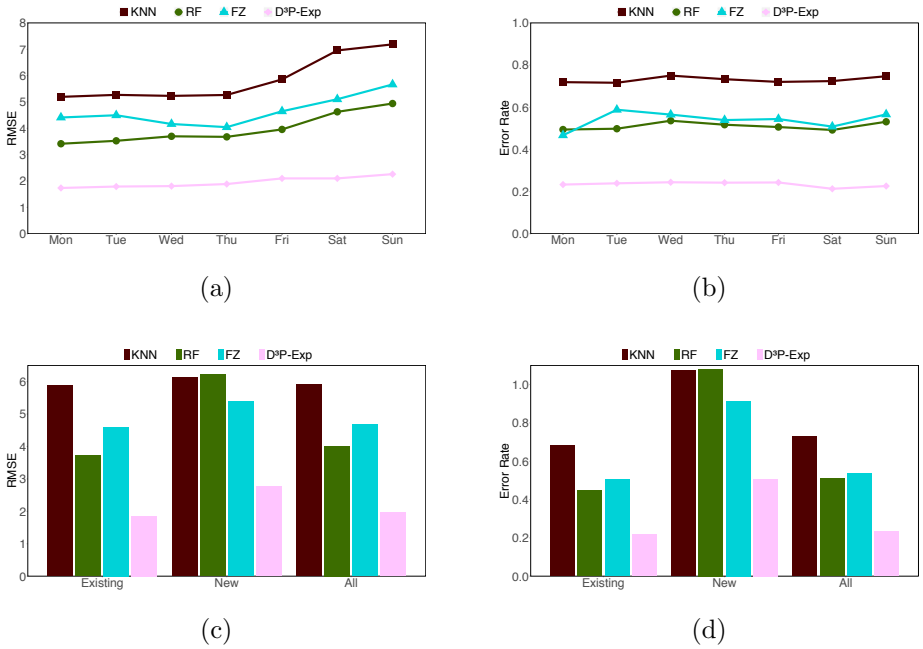
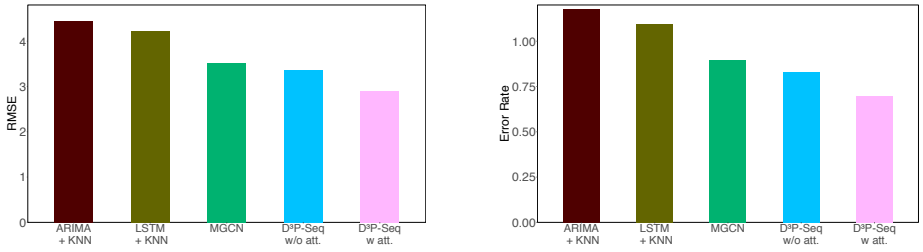


Figure 3.7: Performance on predicting the expected demand. (a) RMSE and (b) ER of all stations across different days in the week. (c) RMSE and (d) ER of existing vs. newly deployed stations vs. all stations averaged over all days of the week.

with the learning rate of 0.001. The networks are trained on a single Titan Xp GPU from scratch. To preserve the temporal dependencies in the data, we partition the data into multiple batches where each of them contains data of consecutive three months. For example, the first batch includes data from January, February and March, while the second has data of February, March and April. For each batch, we use both real and synthetic data ($p = 0$, ..., $p = 1$ as discussed above) from the first two months for training, and the real data of the third month for testing. We train the two branches of our predictor networks separately, where the ground truth labels are obtained from the real world data. We repeat training on all batches and report the best average performance.

3.5.3 Results

Accuracy of Predicting Expected Demand. The first set of experiments evaluate the overall accuracy when predicting the expected demand of stations. Fig. 3.7a and Fig. 3.7b show the RMSE and ER of the proposed approach (D^{3}P-Exp) and competing algorithms over different days of the week. We see that comparing to naive KNN, the random forest based approach (RF) can reduce the RMSE by about 30% while ER by 20%. However, our approach (D^{3}P-Exp) performs significantly



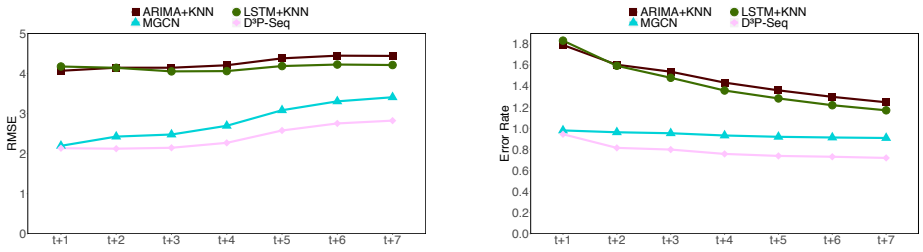
(a) (b)

Figure 3.8: Performance on predicting the instant demand. (a) RMSE and (b) ER of the competing approaches.

better, and can achieve up to three times improvement in both RMSE and ER. In particular, on average the RMSE of D^{3}P-Exp is approximately 1.961, which means when predicting a station's expected demand, the value estimated by our approach is only about ± 2 with respect to the ground truth. This confirms that the proposed approach can effectively model the complex temporal and spatial dependencies within the evolving station network, and exploits that to make more accurate predictions. In addition, we observe that the RMSE tends to increase on weekends compared to weekdays for all algorithms. This is because in practice the absolute demand on weekends is larger, which often leads to bigger RMSE. Note that the ER remains relatively consistent across different days.

Planned vs. Existing Stations. This experiment investigates the prediction performance of different approaches on the planned new stations which haven't been deployed yet, and existing stations which are already been in operation. Fig. 3.7c and Fig. 3.7d show the average RMSE and ER of the proposed approach (D^{3}P-Exp) and the competing algorithms on the planned, existing, and all stations respectively. We see that all of approaches perform better on the existing stations than the planned. This is expected because for existing stations we have access to their historical demand data, which is not available for planned stations. We also observe that although the functional zone based approach (FZ) performs better than the baselines for the planned stations, it fails on the existing stations (performs worse than RF). This is because by design FZ is tuned to predict demand of new stations in the context of system expansion, but not for existing ones. Finally, we see that for both planned and existing stations our approach (D^{3}P-Exp) performs consistently the best. For the planned stations, it halves the errors comparing to the state of the art approach FZ, while for the existing stations, it offers about three-fold improvement over the baselines.

Accuracy of Predicting Instant Demand. This set of experiments evaluates

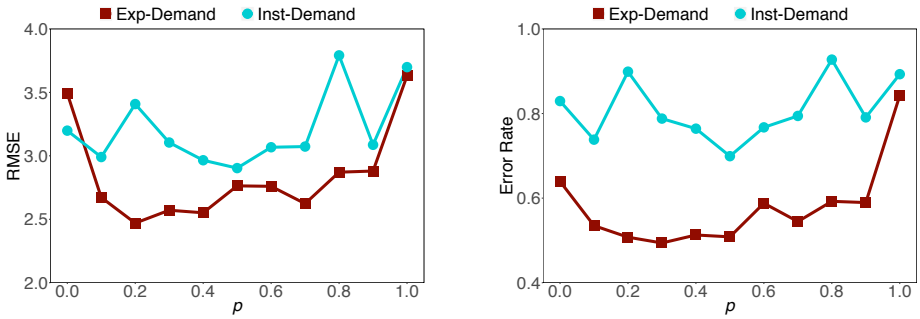


(a) (b)

Figure 3.9: (a) RMSE and (b) ER of the predicted instant demand for different prediction lengths.

the performance of different approaches when predicting the future instant demand. Here we only consider the planned stations, since it is straightforward to predict for the existing stations given their historical data. We ask all approaches to predict the instant demand over the next seven days, and report the average accuracy. Fig. 3.8 shows the RMSE and ER of the proposed approach (D^{3}P-Seq) and the competing algorithms. We see that in this challenging case, our approach (D^{3}P-Seq) can still achieve an average RMSE of 2.903, which is over 30% lower than the baselines (similar gap can be observed in ER). It is also superior to the state of the art MGCN approach which also uses multi-graph GCN, with about 20% reduction in RMSE and ER. This confirms that even for the planned stations without historical data, our approach can still accurately predict their future instant demand within a certain time window. In addition, we find that the attention mechanism in our approach is very effective. Without using attention architecture in the decoder, the performance of our approach drops by approximately 15%, which is still better than the state of the art.

Accuracy vs. Prediction Length. This experiment studies the accuracy of competing approaches when predicting instant demand over different time intervals. As in the previous experiment, here we also only consider prediction performance for the planned station. We vary the length of the prediction time window from 1 to 7, i.e., from predicting the demand of stations on the immediate next day $t+1$, to that on the subsequent seven days $t+7$. Fig. 3.9 shows the RMSE and ER of the approaches under different time windows. We observe that in general, the RMSE increases as the length of the time window grows, especially for our approach (D^{3}P-Seq) and the state of the art MGCN. This makes sense because clearly predicting demand over a longer time window is more difficult. On the other hand, we see that the ER of baselines are higher for short window lengths comparing to the MGCN or our approach. We find that this is because the baselines tend to report random



(a) (b)

Figure 3.10: Prediction performance of our approach trained on datasets with different levels of system expansion dynamics. (a) RMSE and (b) ER.

estimations on the future demand, where for shorter windows this can lead to larger ER, but will be averaged out for longer time windows as the ground truth demand grows in later days. Finally, we see that MGCN can offer comparable performance with our approach (D^3P-Seq) when predicting for the immediate next timestamp. However as the prediction length increases, our approach consistently outperforms MGCN, with a performance gap of up to 26%.

Impact of Different System Expansion Dynamics. The last set of experiments investigates the impact of different levels of system expansion dynamics on the proposed demand prediction approach, and the validity of using the synthetic data for training. As discussed in Section 3.5.2, the reason why we use synthetic data in addition to real data for training is that the real data only represents one sample (or run) of system expansion, which is not sufficient for our models to pick up the general expansion process. Therefore, we synthesise more datasets by randomly selecting a subset of existing stations according to a probability p, and assume those stations as newly deployed. This allows us to generate datasets with different levels of extra injected expansion dynamics. In particular, we vary p from 0 to 1, where $p = 0$ is the real data without any extra dynamics, and $p = 1$ leads to the extreme case where all the stations are considered as newly deployed at each timestamp. Therefore, we train our models with both the real ($p = 0$) and synthetic ($p = [0.1, ..., 0.9]$) data, but in all experiments we evaluated our approach with the real data. As shown in Fig. 3.10, we see that as p increases from zero, our approach tends to make more accurate predictions for both expected and instant demand (lower errors). This confirms the validity of our data augmentation approach, in that by artificially injecting the synthetic dynamics, we essentially force the GCN to learn how to better react to the deployment of new stations. We also observed that for larger p values,

the errors (both RMSE and ER) tend to increase for both types of demand. This is also expected because in those cases the excessive injected dynamics would mute the useful information coming from local LSTMs at individual stations and confuse the GCN, leading to deterioration of performance. Therefore, this means with carefully selected p, the synthetic data can help the learning process to capture the dynamics caused by system expansion, leading to more accurate predictions. Empirically we find that p with values around $0.4 \sim 0.6$ would achieve the desired balance between incorporating the historical information and learning from the expansion dynamics.

3.6 Conclusion

In this chapter, we propose a novel data-driven demand forecasting approach for urban mobility system using a fast expanding electric vehicle (EV) sharing systems as an example, which learns the complex spatial and temporal system dynamics from the continuous expansion process, and is able to robustly predict demand for both existing stations and the planned new stations which haven't been deployed yet. Specifically, the proposed method first encodes the local temporal information at the individual station level, and then fuses the extracted features with a novel Dynamic Graph Convolutional Neural Network (DGCN) to account for the spatial dependencies between different stations. The demand of stations is then estimated by a multi-scale prediction network, which forecasts both the long-term expected demand and the instant future demand of the shared e-mobility system. We evaluate our approach on data collected from a real-world shared e-mobility platform in Shanghai for a year. Extensive experiments on real and synthetic data have shown that our approach consistently and substantially outperforms the state of the art in predicting both the long-term expected and the immediate future demand of the fast expanding system. Our proposed method may help businesses or public operators to reliably evaluate planned system changes and expansions, offering valuable decision support. However, there are also some limitations of the proposed approach. For instance, it does not consider the potential user demand, which is caused by the nature of the real-world dataset. Overall, it helps to improve the shared mobility systems by accurately predict future demand. The next chapter also improve such systems from the infrastructure optimisation perspective.

Chapter 4

Infrastructure Optimisation

4.1 Introduction

In this chapter, we study another important problem for urban shared mobility systems, the optimisation of their infrastructure. In particular, we aim to find the optimal ways of deploying the infrastructure for the shared mobility systems (stations in particular) during their operation, in seek for the desired balance between gains and cost. This has become increasingly important as the services mature: they could have passed the initial fast growing stage in which high deployment cost can be tolerated, but are more keen to fine-tune themselves to increase profit margin, while improving coverage and service level if possible. To achieve that, the shared mobility systems need to carefully select which stations to open or close and when, so that the overall performance is optimised throughout lifespan. Similarly as in the previous chapter, we consider data collected from a real-world shared e-mobility system, but the approaches developed in this chapter can be used for general shared mobility systems in urban settings.

This particular problem falls into a broader class of optimisation tasks in the context of shared mobility systems, which have attracted attentions from various communities. For instance, the work in [10] addresses a similar problem of charger planning for electric vehicle sharing services. It proposes a demand-aware planning approach and uses heuristics to approximate the optimal plan which ensures both pervasive coverage (in terms of reaching as many POIs as possible) and satisfies sufficient user charging demand. However, this approach only performs one-off optimisation relying on aggregated historical demand estimates, which won't be able to adapt as situation changes. On the other hand, the recent work in [50] considers the incremental cases, but essentially it uses greedy-based approaches to re-compute

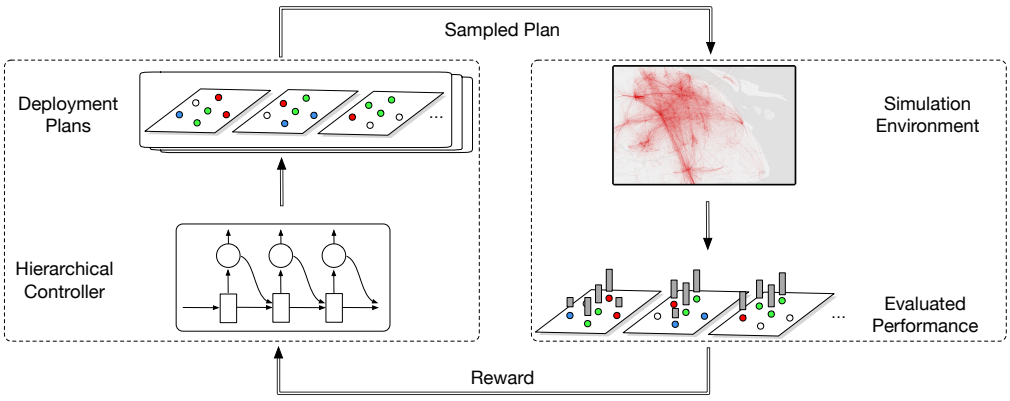


Figure 4.1: Workflow of the proposed hierarchical neural search approach. The hierarchical controller generates possible deployment plans, which are then run in the simulation environment to evaluate their performance. The reward signals (e.g. net revenue and service coverage) obtained are used to update the controller parameters with reinforcement learning.

for charger planning, which may not be efficient in many cases. In addition, most of the existing work aims to maximising certain objectives given fixed budgets, such as service coverage and satisfied demand.

In this thesis, we consider a much more realistic case, where the deployment of the shared mobility system could be optimised dynamically as they operate, with or without specifying concrete budgets. In particular, we would like to find the dynamic deployment plan that guides system deployment through time, which is optimal in terms of given objectives (e.g., service coverage or revenue), and also *self-sustaining*. This requires that the income of the system under the deployment plan at any time should be able to cover the cost of adjusting its station deployment. In practice, such plans could better cater for varying user demand and provide substantial improvement in performance, e.g., we could temporally close stations that are often quiet on specific days (weekdays vs. weekends), while deploying "overflow" stations in the presence of demand surges. However, it is not trivial to find the optimal deployment plans, in that i) for such mobility systems operating at city scales, the search space of station deployment can be prohibitively large; and ii) for a given deployment plan, accurately evaluating its performance could be challenging. This may be easy for static metrics such as service coverage, but for those that involve interactions between users and the systems, e.g., revenue of orders, it is often difficult to evaluate without actually running the systems with the deployment plan. Using historical data to extrapolate is one option (as in [10] and [50]), but as we show later the performance can be limited.

To address the challenges, in this chapter we build a high-fidelity simulation environment for shared mobility systems, which is able to capture the operation details of such systems at fine-granularity. The simulation environment is calibrated with data collected from a real-world shared mobility system for a year, to ensure that it can faithfully simulate the real system behaviors and user interactions in practice. With the simulation environment, we design a novel multi-agent neural search algorithm to address the deployment optimisation problem. Specifically, we consider a hierarchical search structure, which employs a two-level controller to iteratively propose possible deployment plans, and evaluates their performances in the simulation. The results (as rewards) are propagated back to the controller, whose parameters are updated accordingly so it can generate better deployment plans in the future, as shown in Fig. 4.1. Concretely, our contributions are as follows:

- We design a high-fidelity simulation environment for shared mobility systems at city scales. We abstract the key functionalities and operations that are crucial in real shared mobility systems, and calibrate the simulation environment with data collected from the real-world systems. Thus our simulation is able to faithfully capture the fine-grained details of the system operation, which provide support for training and testing the proposed search algorithm.

- We propose a novel multi-agent neural search algorithm to address the deployment optimisation problem. We formulate the problem with multi-agent reinforcement learning (MARL) framework, and develop a novel hierarchical controller architecture, which learns to search for optimal deployment plans efficiently.

- We evaluate the proposed approach extensively, and show that development plans proposed by our approach significantly outperform the actual plan used in the real-world, offering about 30% better net revenue and service coverage. We also show that our search algorithm is superior to the state-of-the-art optimisation approaches, achieving improvements in both net revenue and service coverage.

The rest of this chapter is organised as follows. We discuss additional related work in Section 4.2. In Section 4.4 we present the design of our high-fidelity simulation environment for shared mobility systems. In Section 4.3 we present our multi-agent neural search algorithm to solve the sustainable deployment optimisation problem, and evaluate the proposed approach against baselines in Section 4.5. We conclude this chapter in Section 4.6.

4.2 Related Work

As in the previous chapter, in this section we delve into the existing approaches that are particularly related to the contributions presented by this chapter. For general background we refer the readers to Chapter 2.

4.2.1 Facility Planning and Deployment

Our work is also related to the facility location problem, which has been extensively investigated in many application scenarios, such as finding locations for warehouses [137], chain stores [138], and bike sharing stations [139]. In the EV context, there is also a variety of existing work, focusing on EV specific tasks such as charger deployment. For example, the work in [140] proposes approaches to find locations to deploy EV charging facilities relying on electric taxi trajectory data, while [141] uses a model to optimise the charger distribution across the city. The work in [142] also considers the charging cost of the EV drivers when optimising charger locations, while [50] proposes two heuristic-based algorithms to place EV chargers given their potential social benefits. It also considers the cases of incremental deployment, where they essentially re-run the proposed algorithms on the new set of candidate locations. Our work is different in that we have different objectives to optimise. These work considers the deployment public chargers across the city, which aims to satisfy as much charging demand as possible with a given budget on charger deployment. In our case besides service coverage, we also care about profit as a private EV sharing service provider. In addition, our approach doesn't need to work with a predefined budget, which is often difficult to estimate in practice, but just finds the self-sustaining deployment plans that can cover the cost. On the other hand, the work in [10] also studies EV charger planning for private EV-sharing platforms, which shares the similar problem with our work. However, it assumes the deployment of chargers is an one-off task, and doesn't consider the dynamic deployment cases as investigated in this work.

4.2.2 Neural Search and Optimisation

Recently, there is a emerging interest in using deep neural networks to solve a broad range of optimisation problems. Given their structure, it is often possible to formulate the optimisation problems as sequential decision progresses and use a reinforcement learning agent, which learns the heuristic implicitly to find a solution. For instance, the work in [143] proposes a DQN based approach to address the Max-Cut problem, which becomes the new state-of-the-art. In [144] the authors design

a general RL approach for combinatorial optimisation and show promising results on the Maximum Independent Set Problem. The work in [145] tackles the Graph Coloring Problem for large graphs by using efficient network architectures to speed up optimisation. Besides the traditional optimisation problems, such neural search techniques have also been widely used in the emerging AutoML field, which aims to automate the machine learning pipeline. For instance, the work in [146] introduces neural architecture search, which learns to generate good performing child neural networks. Another work [147] uses similar technique, but rather than searching for network architecture it finds the optimal optimiser. Our work bears close resemblance to those existing work, in that we also try to tackle large search spaces, and the evaluation of the generated representations (in our case the deployment plan) is not trivial. However, in this work we try to solve a different problem in the shared e-mobility context, and propose a novel hierarchical controller architecture to search efficiently, which has not been considered in the existing work.

4.3 Infrastructure Optimisation

In this section, we present the proposed infrastructure optimisation approach, which leverages the simulation environment built in the previous section to search for the optimal deployment strategies. We first describe the infrastructure optimisation problem in Section 4.3.1, and then show how this problem can be formulated as a Multi-agent Reinforcement Learning (MARL) task in Section 4.3.2. In Section 4.3.3 we introduce our hierarchical neural search framework to address this task, and discuss parallel training techniques to speed up learning.

4.3.1 The Deployment Optimisation Problem

Let us assume at time $t = 0$, the simulation world is at the initialisation state, i.e., the shared mobility system has some stations deployed and is ready to operate. This is reasonable since in practice, such systems tend to deploy their first batch of stations before going live to the public. Let T be the length of one simulation episode, i.e., we run the simulator for T timestamps (days in our case) in one training/testing pass. As discussed above, we assume during the T days the stations of the service can be dynamically deployed or removed on a daily basis, i.e., at each t, according to a deployment plan $\boldsymbol{S} = \{S_t\}$, $t \in [1, T]$, where S_t is the set of stations to be active at t, drawing from the candidate pool $\mathbb{S}$.

Let $\mathcal{P}(\boldsymbol{S})$ be the set of all possible deployment plan of length T for this simulated mobility system. Then our deployment optimisation problem is to find

the optimal deployment plan $\boldsymbol{S}^* \in \mathcal{P}(\boldsymbol{S})$:

$$\begin{aligned} \boldsymbol{S}^* = \ & \underset{\boldsymbol{S}\in\mathcal{P}(\boldsymbol{S})}{\operatorname{argmax}} \, \texttt{SC}(\boldsymbol{S}) + w\texttt{PM}(\boldsymbol{S}) \\ \text{s.t:} \quad & c(S_t) \leq \texttt{GMV}(S_t), \text{ where } S_t \in \boldsymbol{S}, \forall t \in [1:T]. \end{aligned} \tag{4.1}$$

Here $\texttt{PM}(\boldsymbol{S})$ is the *profit margin ratio* of the deployment plan $\boldsymbol{S}$, i.e., the percentage of profit (net revenue NV) with respect to the total income (GMV) if deploying stations as instructed by $\boldsymbol{S}$. $\texttt{SC}(\boldsymbol{S})$ is the *service coverage ratio* of the deployment plan $\boldsymbol{S}$, which is defined as the normalised sum of two ratios: i) the percentage of satisfied user demand, and ii) the percentage of POIs covered by the service. We assume a POI is covered by a station if it is within $1km$ radius of the station. Therefore, at time t the service coverage of the corresponding deployment snapshot $\texttt{SC}(S_t)$ depends on both satisfied demand and covered POIs. Similar to $\texttt{NV}(\boldsymbol{S})$ defined above, we define $\texttt{SC}(\boldsymbol{S})$ as the mean service coverage of this particular deployment plan over T timestamps, $\texttt{SC}(\boldsymbol{S}) = T^{-1}\sum_{t=1:T}\texttt{SC}(S_t)$. In our case, both objectives are in percentage, and thus we can use a weight $w \in [0, 1]$ to balance them, which can be tuned for scenarios with different priorities, e.g., aiming to roll out service to more users vs. obtain more profit. The constraint requests that the deployment plans should be self-sustaining, in the sense that the income of the system at any t should be able to cover the cost of deployment, i.e., we want the system to be break-even.

In fact, if we unfold the deployment plans over time, this deployment optimisation problem is essentially a constrained combinatorial optimisation problem. However, the search space in our case can be prohibitively large, e.g., with $4k$ candidate stations over 30 days period there will be $2^{4k^{30}}$ possible plans to evaluate. In addition as discussed in previous sections, the impact of deploying/removing a particular station is often complicated, i.e., we won't be able to directly estimate the gain of such an action (e.g., increase in NV and SC) without running the system through. This makes most of the existing heuristics-based optimisation approaches [10, 50] infeasible, as they assume the benefits/utility (e.g., of deploying a particular charger/station) are independent and known a priori. In the next section, we show how we formulate this problem as a Multi-agent Reinforcement Learning (MARL) task, which allows us to explore the search space via trial-and-error.

4.3.2 Deployment Optimisation as a MARL Task

We consider a set of autonomous agents, each of which interacts with the common environment to improve its behavior. Comparing to the single agent settings, the

multi-agent formulation is more suitable to address our problem, because multiple agents could better exploit the decentralized nature of the station deployment task. By design learning can be more efficient than the single agent case, as the action spaces of the agents are much smaller, and the computation can be accelerated by parallel processing. Under this setting, we model the deployment optimisation problem as a Markov Game $G = (N, \mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, in which at most N agents interact with the environment, making decisions as to where and when stations should be deployed. $\mathcal{X}$ represents the states of G and $\mathcal{T}$ is the transition function between states. $\mathcal{A}$ is the joint actions of the agents, $\mathcal{R}$ is the reward function, and γ is the discount factor.

Agents. In our multi-agent formulation, we assume an agent controls the deployment of stations within a certain geospatial region. Essentially, we delegate the agent to manage the station deployment process of that region, who decides which candidate stations should be deployed or closed and when. We partition the space into regions belong to different agents by clustering candidate stations locations in $\mathbb{S}$. In particular, we cluster the candidate stations based on their pairwise distances, where boundaries of regions are formed by convex hulls of clusters. Let us assume we have obtained N regions as the result of clustering, managed by N agents.

To make sure that the agents should face similar learning load, we also require each cluster to have the same amount of M candidate stations. In practice, this may lead to regions with different sizes due to the spatial variation in the density of candidate station locations, e.g., we observe that the obtained regions in city center is much smaller than those at the city edges. However, we found that this won't affect the agents' performance, since the candidate station density is also highly correlated with POI density and distribution of potential user demand. This means although managing regions with different sizes, the agents tend to learn to cover similar levels of demand/POIs with its deployment plans, only at different spatial scales. It is also worth pointing out that our approach can also work with the other space partition paradigms, such as hexagonal or rectangular grids, as long as each region/grid is managed by a individual agent.

States and Observations. At time t, for the i-th region, its state x_t^i encodes information about the user demand, vehicle distribution and station deployment within the boundary of this region. In particular, we consider both the set of currently online stations, and candidate stations that are not yet deployed. For each of them, we include its location `loc`, number of parking spaces `#c`, the deployment cost and the numbers of vehicles parked in the stations at each simulation step within t, as well as the potential future rent/return requests and the average value of

potential future orders in the next timestamp. The global state $\boldsymbol{x}_t$ is the combination of states of each region $\boldsymbol{x}_t = \{x_t^i\}$, $i \in [1, N]$. Let agent i manage the i-th region. At time t, we assume the agent observes its local state $\{x_t^i\}$ as well as the global state $\boldsymbol{x}_t$. This allows the agents to observe and interact with not just its local environment, but also learn to better cooperate with the others that have connections with them, e.g., agents of neighbouring regions, or those with high correlations in terms of user demand (i.e., users may frequently travel from one region to another).

Agent Actions and State Transitions. For agent i, its action a_t^i describes the deployment snapshot of the i-th region, i.e., which stations should be deployed or removed. Therefore at time t, a joint action $\boldsymbol{a}_t \in \mathcal{A}_t^1 \times ... \times \mathcal{A}_t^{N_t}$ all the N agents forms the complete deployment snapshot S_t, as introduced above. Upon performing the joint action at t, the current state $\boldsymbol{x}_t$ will transit to the next state $\boldsymbol{x}_{t+1}$ according to the state transition probabilities $\mathcal{T}$, which are defined as $\mathcal{T}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{a}_t)$. Note that in many MARL cases, it is often not possible to describe the state transitions as functions with analytical forms, and thus in our case we don't attempt to model $\mathcal{T}$ explicitly, but rely on our simulator to capture the state transitions.

Reward Function. For each agent, the reward r_t^i of taking an action at time t is determined by the reward function:

$$\mathcal{R}^i(\boldsymbol{x}_t, \boldsymbol{a}_t) : \mathcal{X} \times \mathcal{A}_t^1 \times ... \times \mathcal{A}_t^{N_t} \to \mathbb{R} \tag{4.2}$$

As discussed above, in our deployment optimisation problem, we would like to maximise both the profit margin (`PM`) and the service coverage (`SC`). In practice, we found that the two objectives often diverge, e.g., maximizing `PM` would often lead to greedy agents that always try to deploy new stations at "hot" locations with high profit. On the other hand, optimizing `SC` tend to encourage the agents to spread stations across the space, while keeping many less profitable stations active, resulting in decrease in net revenue. In addition, we also require the service to be sustainable, in that the GMV of the deployed stations should be able to cover their cost. To balance such factors, given the agent action a_t^i, which requires the set of stations S_t^i to be active at time t, we design the reward function r_t^i based on our optimisation objectives as in Eq. (4.1):

$$r_t^i = g^{\texttt{SC}}(S_t^i, S_{t-1}^i) + w\, g^{\texttt{PM}}(S_t^i, S_{t-1}^i) + \lambda \min\{\texttt{NV}(S_t^i), 0\} \tag{4.3}$$

where $g^{\texttt{SC}}(S_t^i, S_{t-1}^i)$ and $g^{\texttt{PM}}(S_t^i, S_{t-1}^i)$ calculate the *improvement rate* in terms of service coverage `SC` and profit margin `PM`, given the current deployment snapshot S_t^i and the previous one S_{t-1}^i. w is the weight balancing the two objectives, as in

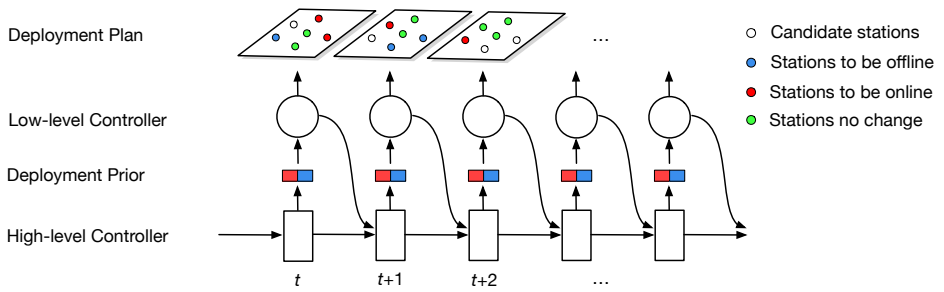


Figure 4.2: The architecture of the proposed hierarchical controller.

Eq. (4.1). The term $\lambda \min\{\texttt{NV}(S_t^i), 0\}$ penalizes any S_t^i that produces negative net revenue with a scaling factor λ, which is learned via grid search. Given the reward function, each agent aims to maximise its discounted reward $\mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}^i]$, where $\gamma \in [0, 1]$ is the discount factor.

4.3.3 Hierarchical Neural Search

Under the above MARL formulation, we design a novel hierarchical neural search algorithm, which learns to generate better deployment plans over time. The key idea is to use a controller (in the form of neural networks) to generate various deployment plans (i.e., actions), and evaluate the performance of those plans in our simulation environment. The reward/penalty signals collected from the simulation are then propagate back and used to update the controller with reinforcement learning paradigm, so that it tends to produce improved plans in future runs.

Hierarchical Controllers. In this work, we consider a LSTM-based controller, which recursively generates deployment plan S_t at each t for T timestamps. In fact, given the multi-agent setting, for each agent i, we can use a boolean string of length M to represent S_t^i, where M is the total number of candidate stations in the regions. Such encoding has been widely considered in tasks such as searching for neural network architectures. However unlike the existing work, in our case even with multi-agent formulation, directly generating and searching from the 2^M sized encoding space is prohibitively expensive. To tackle this challenges, we design a hierarchical controller which generate S_t^i progressively, as shown in Fig. 4.2. The key idea is that instead of directly sampling from the distribution of deployment plans at each t, we generate S_t^i in two stages. Concretely, we design a hierarchical structure with two connected controllers: a *high-level* and a *low-level* controllers respectively.

High-level Controller Design. Given the previous and current states, rather than

delving into deployment details, the high-level controller only generates a categorical *deployment prior* for the region i at time t. In particular, we consider a deployment prior as a pair (S_t^{i+}, S_t^{i-}), where S_t^{i+} is a categorical variable indicating the amount of new stations to be made online during time t at region i, e.g., 10% of the total candidates in the region, while S_t^{i-} is for the stations to be made offline. In this case, if the domain of the variables is limited, then for the high-level controller the search space at each timestamp t is manageable. On the other hand, such a deployment prior indicates the coarse volume of the desired deployment plan for the region at time t, e.g., the high-level controller may prefer to deploy more rather than removing in some cases since it predicts demand would surge.

Low-level Controller Design. Now we explain the design of the low-level controller, which translates the received deployment prior into the concrete deployment plan S_t^i. Essentially, now the low-level controller needs to find two sets of stations, one to deploy and one to close, while the cardinalities of them are known. In this case, one straightforward way is to continue to sample the constrained search space which is already much smaller, and generate the deployment plan S_t^i accordingly. However, we found that in practice this would still lead to many meaningless trials without converging. Therefore, in this work we adopt an efficient approach to generate S_t^i. Concretely, for each candidate station s_j^i within region i, we compute a score which is the combination of two values: i) its service coverage, i.e., the ratio that it could contribute to the total service coverage in this region, and ii) the ratio of expected demand it could satisfied with respect to the total demand in region i. Here we use the forecasting approach discussed in the previous chapter to predict the expected demand for s_j^i at t, given the previous states in this simulation episode.

Now we obtain the ranked list of the candidate stations in region i based on their scores. Intuitively, we should deploy the best station candidates if they are not yet in operation, and close the poorly performing ones. To generate such S_t^i at t, our low-level controller adopts a ϵ-greedy approach for both deploying and closing stations. In particular, it iteratively chooses the current best/worst station candidates with the probability $1 - \epsilon$, while making random selections for the rest, until the required number is reached. In our experiments, we set $\epsilon = 0.1$. In this way, the low-level controller balances exploration and exploitation in searching of S_t^i, which makes our algorithm more robust to noises and perturbations.

Training. We consider a shared-weights controller for our agents, which is trained with reinforcement learning. As discussed in Section 4.3.2, the generated deployment plan $\boldsymbol{S} = \{S_t\}$ is a sequence of actions $\{\boldsymbol{a}_t\}$ $(t = 1 : T)$ to adjust the vehicle sharing service. By applying $\boldsymbol{S}$ in the simulation, we could evaluate the performance of $\boldsymbol{S}$,

and compute the reward as Eq. (4.3) and Eq. (4.2). The reward signal is then fed back to the controller, to help updating its parameters θ. Therefore the updated controller would have high probabilities to propose better deployment plans. Note that here θ is only relevant to the high-level controller, as the parameters of the low-level controller (the GCN) have already been trained. In our case, the reward structure is not differentiable (min in Eq. (4.3)), and thus we consider policy gradient approaches to iteratively improve θ. In our case, we train the controller to maximise the performance of its sampled deployment plans $\boldsymbol{S}$. The training objective can be formulated as follows:

$$J(\theta) = \mathbb{E}_{p(\boldsymbol{S};\theta)}[\mathcal{R}(\boldsymbol{S})] \tag{4.4}$$

where $\mathcal{R}$ is the reward collected from simulation, and the expectation is taken over $p(\boldsymbol{S};\theta)$, i.e., the distribution of deployment plans $\boldsymbol{S}$ given θ. In this work we use the Proximal Policy Optimisation (PPO) approach [148] to optimise $J(\theta)$, which is more sample efficient than the standard REINFORCE [149], and offers faster convergence of the controller. For the sake of completeness and clarity, here we refrain from introducing the formulation details of PPO, which will be discussed in Chapter 5, to show how we tailor the standard PPO approach for our specific tasks. We refer the reader to Section 5.4.2 for more details.

4.4 Simulating Shared Mobility Systems at City Scale

In this section, we present our design of the high-fidelity simulation environment for shared mobility systems at city scale, which builds the playground for the proposed multi-agent neural search algorithm to learn, by trial-and-error, how to optimise deployment with specific goals (e.g., revenue vs. service coverage) in mind. In the previous chapter, we have already introduced some key concepts of characteristics of the shared mobility system (Section 3.3.1). In the following we provide a more systematic overview of such systems (Section 4.4.1), and describe how our simulation environment handles deployment dynamics (Section 4.4.2) and user demand (Section 4.4.3), as well as its operational models (Section 4.4.4) and calibration (Section 4.4.5).

4.4.1 Overview of Shared Mobility Systems

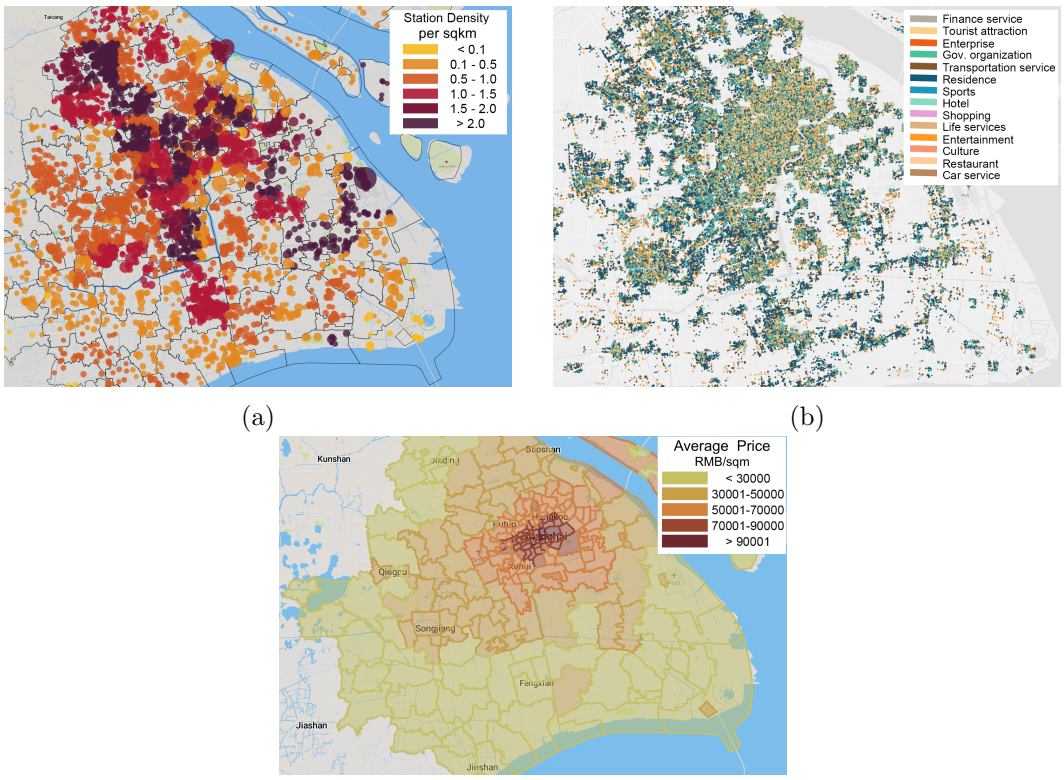
The Shared Mobility Model. As mentioned in the previous chapter, in this thesis we consider a *station-based* shared mobility services, i.e., the users can only pick up vehicles from and return them to the available stations operated by the

vehicle sharing service. Let s be such a station. In our case, s can be represented as a tuple $(\texttt{loc}, \#\texttt{c})$, where `loc` denotes the location (e.g., latitude and longitude coordinates) of s, and $\#\texttt{c}$ is the total number of parking spaces within the station s. In practice, the system may deploy new stations or close existing ones across the city. When a station s is firstly deployed for operation, a number of vehicles (denoted by $\#\texttt{v}$) would be assigned to s, where typically $\#\texttt{v} < \#\texttt{c}$. This means at the beginning the station s should have $\#\texttt{v}$ vehicles available for pick up and $\#\texttt{c}$ - $\#\texttt{v}$ free spaces for potential returns.

Particularly for shared e-mobility systems, we assume the vehicles are of limited range depending on different models, e.g., in ideal conditions a Rover E50 typically has the range of $100km$ while a BMW i3 could cover $180km$. In this thesis, we assume the fully charged ranges of the EVs for given vehicle models are fixed, and during normal driving the remaining range can be determined by their discharging curves [150]. In practice, we observe that the time for charging of the EVs is much longer than refilling the traditional vehicles, which however can be estimated by the corresponding charging models [150], given the remaining range, battery capacities and charger specifications.

Given the set of its available stations S, the shared mobility system operates as follows. Assume that at time t, a user wants to rent an vehicle from a certain station $s^{\mathrm{o}} \in S$, and return to the destination station $s^{\mathrm{d}} \in S$. If she finds there is at least one vehicle available at the picking up station s^{o} (in e-mobility systems the vehicle has to be sufficiently charged with enough range to cover her planned trip), she will post an order $o_t = (s^{\mathrm{o}}, s^{\mathrm{d}})$ to the system. Upon accepting the order, the system allows the user to take over the vehicle. The order is considered to be completed (i.e., the demand of this user is satisfied) when the vehicle is returned to the destination station s^{d} (and charged in the EV context). The price for this order is calculated based on her rental duration.

The Shared Mobility Data. We consider the same sets of data as in the previous chapter. In particular, the dataset contains a) the complete *transactions of orders* in the system, i.e., when and where a user rented/returned a vehicle, and b) information on *station deployment*, i.e., when and where a station was deployed or closed. In the order data, each record contains detailed information about this order, including the anonymous user ID who initiated the order, the ID of the origin/destination stations, the timestamps of pick-up/return, the total duration of the order and the final price etc. In total we have collected >7 million valid order transactions, which were generated by approximately 0.4 million active users during the one year period. For the station deployment data, we collected the service status at each



(a) (b)

(c)

Figure 4.3: Visualisation of the collected data. (a) The distribution, density and sizes (# of parking spaces/charging docks) of the stations in the real-world shared mobility system. (b) POI distribution in the city. (c) Average property price across different regions in the city.

day during the 12 months period, including station locations, numbers of parking spaces (charging docks) in each station (see Fig. 4.3a for visualisation), and when and where stations were deployed/closed. We also collected additional auxiliary data for our simulation, such as the POI data of the city [128] (see Fig. 4.3b for visualisation), the average property prices across different regions [151] (see Fig. 4.3c for visualisation), and taxi trajectory data during the same period of time [152].

4.4.2 Simulating Dynamic Deployment of Stations

We construct our simulation environment by first simulating the deployment in the stations of the shared mobility system. As discussed in the previous section, in practice a station can be viewed as a tuple $(\texttt{loc}, \#\texttt{c})$, where `loc` is its location coordinates, and $\#\texttt{c}$ is the total number of parking spaces within the station. For simplicity, in our simulation we assume that any stations to be deployed are drawn

from a candidate pool $\mathbb{S}$, containing all possible candidates of stations in the system. Without loss of generality, we consider a 2D world where a station candidate $s \in \mathbb{S}$ is a point at location `loc` and with #`c` parking spaces. In our implementation, we generate the candidate pool $\mathbb{S}$ according to the station deployment data collected from the real-world system, i.e., for any station s that has been deployed once in that system, regardless of whether it had been closed or not, we add s to the pool $\mathbb{S}$. In this way we obtain the candidate pool of stations for deployment in simulation, which forms the search space for the proposed approach.

In our simulation, we consider *dynamic deployment* of stations, i.e., stations can be opened or shut down at arbitrary times. Without loss of generality, we assume the time granularity for station deployment is days, i.e., they can be adjusted on daily basis. Let $S_t \subseteq \mathbb{S}$ be a subset of the candidate pool. We refer to S_t as a *deployment snapshot* at time t, and assume by the end of the timestamp only the stations in S_t will remain active in simulation. The sequence of such deployment snapshots from the beginning to the end of the simulation episode is defined as the *deployment plan* $\boldsymbol{S}$, where $\boldsymbol{S} = \{S_t\}$, $t \in [1, T]$. In essence, $\boldsymbol{S}$ describes the evolving process of the stations in the shared mobility system, which as discussed later is the key for performance, such as the revenue and demand satisfaction rate of the service.

We also assume that for each candidate station s, there is a cost $c(s)$ for being active in operation of the system per unit time. In our simulation, this cost is set based on two factors: i) the property price around s; and ii) the size (i.e. number of parking spaces #`c`) of s. This is consistent with the cases we observed from the real-world data, where the rental prices of stations depend on their sizes, e.g., those with more parking spaces tend to be more expensive, and also differ in different areas. Therefore, at given time t the cost of running the service depends on the current deployment snapshot S_t, i.e., $c(S_t) = \sum_{s \in S_t} c(s)$. Then for a deployment plan $\boldsymbol{S}$, we define the total cost in this episode $c(\boldsymbol{S})$ as the sum of the costs incurred by its snapshots: $c(\boldsymbol{S}) = \sum_{t=1:T} c(S_t)$. Note that here for simplicity we don't consider the upfront deployment cost such as installing chargers etc., which in practice can be factored into the long-term running cost, and assume there is no overhead of shutting down or re-opening stations.

4.4.3 Generating Spatio-temporal User Demand

To better simulate the operation details of the actual shared mobility system, we set 10min as one simulation step. Note that this is different from the timestamps considered in station deployment process as discussed above, which is a day (24

hours), containing 144 simulation steps. With such settings we are able to simulate at much finer granularity, e.g., we could generate each individual user demand for the mobility sharing service, and observe how it is satisfied or why not, rather than relying on aggregated information.

We define a particular user demand as a tuple $d = (k, \mathtt{o}, \mathtt{d})$, where k is the current simulation step, $\mathtt{o}$ and $\mathtt{d}$ are the origin and destination of this demand respectively. We represent $\mathtt{o}$ and $\mathtt{d}$ with 2D coordinates e.g., GPS locations across the space. This represents the intention of travel from $\mathtt{o}$ to $\mathtt{d}$ of the users, and in our simulation we learn to generate user demand from the real-world data. In particular, we consider the historical order transactions of the shared mobility system, as well as the taxi trajectory data (cleaned to only have origins and destinations). We use Gaussian Processes (GP) [59] with RBF kernels to approximate the spatial-temporal distributions of the user demand. In particular, we first consider the pick up demand (i.e., the origins $\mathtt{o}$) and learn GP models over space and time, with which at each simulation step k we can obtain a spatial distribution of the potential origins in user demand. This also means for any point in the 2D space, we have an estimate of the numbers of users that intend to travel from that point at step k, with certain confidence interval obtained from the GP.

When generating the demand in simulation, at given step k we sample the learned GP over the candidate station pool $\mathbb{S}$, i.e., at each candidate station we obtain the amount of travel demand starting from that location at step k. Here we only sample over $\mathbb{S}$ rather than the entire 2D space because eventually user demand will be satisfied at those stations, while the GP model has already taken the neighbourhood knowledge into account, e.g., nearby taxi demand or vehicle pick-up demand at stations within certain distance. Therefore in the following text, we consider the user demand only at the candidate stations, i.e., $d = (k, s^o, s^d)$, where $s^o, s^d \in \mathbb{S}$.

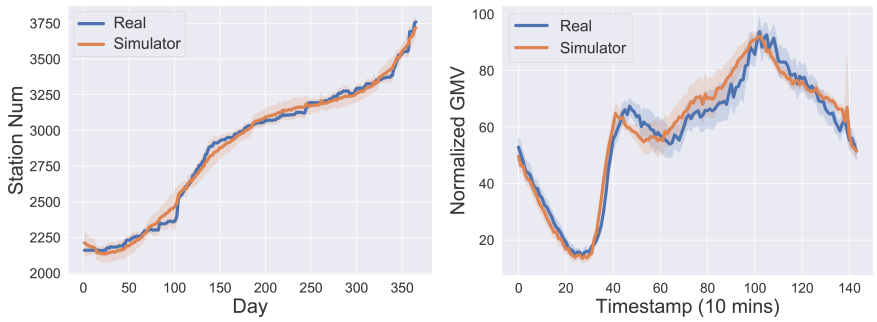
To generate the complete demand in pairs of origins and destinations, for a user demand originated from $s^o \in \mathbb{S}$, we use another GP to approximate the spatio-temporal distribution of possible destinations from the historical orders of the mobility system as well as taxi data. In essence, at step k for this demand with origin s^o, the GP model generates a 2D distribution over the space, and by sampling from the distribution over the candidate pool $\mathbb{S}$ we can pin down the destination station s^d for this particular demand. The reason why we adopt this two-phase approach rather than sampling independently from the distributions of origins and destinations, is that in practice the origins and destinations of user demand are highly correlated, and the correlations often depend on different times. In addition,

unlike existing work [10] which directly uses historical data as demand, we use GPs to i) fuse different sources of information (taxi and the mobility data), and ii) generate distributions of user demand rather than scalar values for robustness. As shown later in Section 4.5, this improves the fidelity of our simulation, and helps the proposed algorithm to generalize better.

4.4.4 Operating Shared Mobility Systems in Simulation

For each simulation episode, let $\boldsymbol{S}$ be a deployment plan of the vehicle sharing stations, e.g., generated by our search algorithm. In essence, $\boldsymbol{S}$ describes which candidate/existing stations should be deployed/removed from the service and when. We assume that this station deployment/adjustment process happens at the beginning of a timestamp t (0am on each day), which updates the active stations according to the corresponding snapshot S_t in the deployment plan $\boldsymbol{S}$. Note that any candidate station to be deployed, we allocate certain number of vehicles (fully charged) to this new station for its future operation, according to its maximum capacity (#c) and a probability ratio learned from the historical data. Typically in our simulation we assign 0.5#c to 0.7#c of vehicles to the new stations, leaving sufficient parking spaces for incoming vehicles from other stations.

Then for each simulation step k (there are 144 steps per time t), our simulator generates user demand as discussed in Section 4.4.3. Concretely, let $d = (k, s^o, s^d)$ be one demand generated at step k, indicating that a user would like to travel from the candidate station location s^o to s^d. This demand could be accepted by the shared mobility service only if i) both candidate stations s^o and s^d are online, i.e., active in operation; and ii) at the origin station s^o, there is at least one vehicle to cover the trip to s^d. Once the demand is accepted, our simulator creates an order, assigns an vehicle to serve this order, and estimates the order duration (rental time in terms of simulation steps) from the model learned from historical data. When this order is about to finish, i.e., at the end of the pre-defined simulation step, our simulator checks the status of the destination station s^d, and the order is announced to be completed only if there is available parking space at s^d. If not, it chooses a nearby station for vehicle return. The income of this order is computed based on the rental time. Otherwise the order is considered to be failed, where there will be no income for the service. Therefore at time t, with the deployment snapshot S_t, we define the total revenue value (i.e., Gross Merchandise Value) at t as $\texttt{GMV}(S_t)$, which is calculated over all the satisfied orders of the mobility sharing service during that time. The net revenue value $\texttt{NV}(S_t)$ is then defined as GMV subtracts the deployment cost: $\texttt{NV}(S_t) = \texttt{GMV}(S_t) - c(S_t)$. For deployment plan $\boldsymbol{S}$, we define its GMV and



(a) (b)

Figure 4.4: Simulation calibration. (a) Simulated vs. real station network expansion for one year (averaged over 10 runs), and (b) Simulated vs. real gross Merchandise Value (GMV).

net revenue as their summations over T timestamps: $\texttt{GMV}(\boldsymbol{S}) = \sum_{t=1:T} \texttt{GMV}(S_t)$, and $\texttt{NV}(\boldsymbol{S}) = \sum_{t=1:T} \texttt{NV}(S_t)$ respectively.

4.4.5 Calibrating Simulation Environment

We calibrate our simulation environment with the real-world data as discussed above. As shown in Fig. 4.4a, the patterns of simulated system expansion are very close to the actual expansion carried out during the year, with Pearson correlation 0.9957 and p value $p <$ 1e-10. For demand generation, we use the calibrated system expansion dynamics, and further tune the simulator with respect to the Gross Merchandise Value (GMV), indicating the total revenue of the system. Fig. 4.4b shows that the simulated order data has very similar properties in GMV with the real data, where the Pearson correlation between simulated and real GMV is 0.9599 with p value $p <$ 1e-10.

4.5 Evaluation

In this section, we evaluate the proposed multi-agent neural search algorithm extensively with the simulation environment as described in Section 4.4. We first explain our experimental settings and competing approaches in Section 4.5.1, and report the results in Section 4.5.2.

4.5.1 Experimental Setup

We compare the proposed multi-agent neural search (**MANS**) approach with the following baseline approaches:

- **Fixed Deployment (FD)**, which keeps the stations at initialisation unchanged, and there is no stations deployed/removed during the simulation.
- **Human Deployment (HD)**, which uses the deployment plan obtained from the real shared mobility data. Particularly, we use an independent validation dataset which hasn't been considered in simulator calibration, and take the actual deployment plan which had been implemented in the real-world.
- **Revenue-greedy Deployment (REV)**, which selects the top/bottom stations to deploy/remove based on their historical averaged net revenue. The amount of stations to be deployed/removed is obtained by sampling a distribution learned from the validation set as in HD, i.e., the scale of changes in stations is similar with HD, but the decisions on deployment are revenue-greedy.
- **Coverage-greedy Deployment (COV)**, which is also heuristic-based like REV, but deploys/removes the stations according to their service coverage, i.e., it prefers to deploy stations that tend to satisfy more demand and cover more POIs.
- **One-time optimisation (OO)**, which is our implementation of the the state-of-the-art charger location optimisation algorithm in [10]. It assumes that the average user demand at each station can be estimated from historical data, and tries to find an one-time optimal station deployment snapshot that maximise both POI coverage and the demand satisfied rate.
- **Incremental optimisation (IO)**, which is the incremental version of OO (similar to the work in [50]). Unlike OO which only optimises station deployment once and uses the solution throughout the episode, this IO performs optimisation and finds the best set of stations to maintain at each timestamp.

All the competing approaches are implemented with TensorFlow 1.14.0, and trained with NVIDIA 2080Ti GPUs. We set the same initialisation of deployment snapshot S_0 to all the competing approaches, i.e., they start from the same situation, and try to learn the optimal deployment plan from there. As discussed in Section 4.3.3, we consider shared-weights controller for our agents, and train in parallel using the multi-simulation technique to improve efficiency. We evaluate the competing approaches against the following metrics:
Service Coverage (`SC`) as defined in Section 4.3.1, which is the combination of satisfied demand rate and POI coverage; and

Table 4.1: Overall Performance of deployment optimisation by different approaches. ΔSC and ΔNV are obtained with respect to Human Deployment (**HD**). For our MANS, w=1/9 prefers rewards in SC, while w=9/1 reward more on NV. Best performance values are shown in red, and second best in blue.

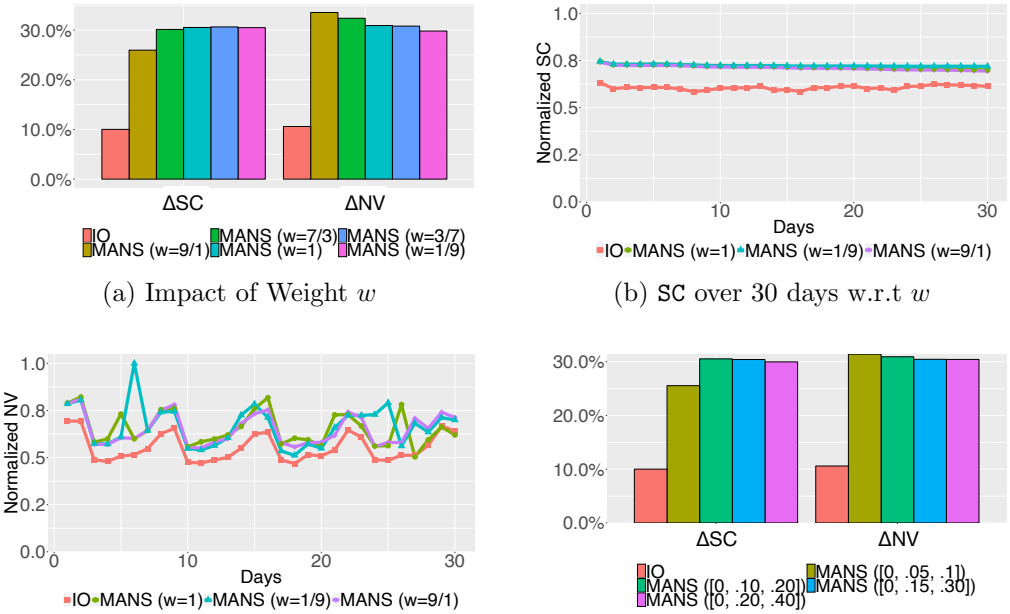
	FD	**HD**	**REV**	**COV**	**OO**	**IO**	**MANS** (w=1)	**MANS** (w=1/9)	**MANS** (w=9/1)
SC	0.5511	0.5529	0.5680	0.6113	0.6054	0.6063	0.7194	0.7192	0.6942
ΔSC	–	–	3.07%	10.92%	9.85%	10.02%	30.53%	30.50%	25.97%
ΔNV	–	–	11.49%	2.49%	9.86%	10.60%	30.92%	29.82%	33.55%

Net Revenue Value (NV), which is calculated as the GMV generated by the deployment plan subtracts the cost on deployment, as defined in Section 4.4.4.

To be fair, instead of directly comparing SC and NV of the competing approaches, we report the improved percentages of SC and NV with respect to the baselines **HD**, indicating the performance gap between the searched plans and the actual deployment process conducted by human.

4.5.2 Results

Overall Deployment optimisation Performance. This set of experiments study the overall performance of the competing approaches. The results are shown in Table. 4.1. Firstly, we see that there is very little improvement in both SC between **HD** and **FD**, which means that the actual deployment plan doesn't offer much benefit. This is as expected because the human knowledge is only applicable to that particular case, but can't generalize to unseen scenarios in simulation. We also see the two greedy-based approaches **REV** and **COV** manage to improve NV and SC respectively, but for the other metrics which are not explicitly optimised, the improvements are marginal. On the other hand, the one-time optimisation approach **OO** strikes a better balance, offering about 10% improvement in both NV and SC. This is because **OO** uses a demand-aware heuristics, which jointly optimises the POI coverage and the amount of demand satisfied, leading to better SC while also indirectly improving NV. The incremental optimisation approach **IO** further improves the performance, in that it is able to dynamically optimise deployment plan over time, but the gap with **OO** is not significant. This is because that essentially it just re-runs **OO** at each timestamp, while its knowledge about the demand is still estimated from historical data, in the same way as **OO**. Comparing to the state-of-the-art **IO**, our approach **MANS** (with w=1) offers about 20% improvement in NV and in SC. The gain comes from: i) we directly optimise NV in our reward structure, and ii) instead of relying on estimates from historical data which is essentially to optimise on a proxy task, we directly learn by trial-and-error on the target task



(a) Impact of Weight w

(b) SC over 30 days w.r.t w

(c) NV over 30 days w.r.t w

(d) SC/NV vs. Action Scales

Figure 4.5: Performance of the proposed MANS approach under different parameter settings. Results of the IO algorithm is included here as the baseline.

using deep reinforcement learning.

Deployment Behaviours under Different Rewards. In this set of experiments, we investigate the impact of the weight w between the two terms in our reward structure, as discussed in Eq. (4.3). Here we vary w in the range [9/1, 7/3, 1, 3/7, 1/9], which gives different importance to the two reward terms $g^{\texttt{SC}}$ and $g^{\texttt{PM}}$. We train our controller under different w, and report the performance of those variants in Fig. 4.5a as well as Table. 4.1. We see that as we decrease the weight, NV drops while SC increases. This is expected, as we essentially give more reward on SC, while ignoring actions that lead to high NV. On the other hand, we also see that SC is less sensitive to w than NV. We see that the increase in SC is very small while w decreases from 7/3 to 1/9. This is because that SC cares about the ubiquity of the service, and thus once the stations in the deployment plan can cover the critical mass of the candidate stations, SC tend to be stable. This can also be observed from Fig. 4.5b, where we show the 30 day trend of SC for $w = [9/1, 1, 1/9]$. We see that under different deployment plans SC remains relatively flat. However, the NV is more sensitive, as shown in Fig. 4.5c. We see clearly that if we reward $g^{\texttt{PM}}$ heavily, the NV on certain days is much higher. It is possible that under such settings, our controller learns to deploy stations temporarily at hot spots on particular days, serving substantial amount of high-value orders. This may bring extra income but

won't necessarily increase the service coverage, since the POIs may already be well covered by neighbouring stations.

Performance vs. Deployment Variability. Finally in this set of experiments, we evaluate the performance of our approach with respect to different level of deployment variability considered in our controller. As discussed in Section 4.3.3, our approach uses a hierarchical controller where the high-level controller generates a categorical *deployment prior* (S_t^{i+}, S_t^{i-}), determining roughly how many new/existing stations should be deployed/removed for each category. Here we vary the magnitude of this deployment prior, and consider four *action scales*: [0, 0.05, 0.1], [0, 0.1, 0.2], [0, 0.15, 0.3], and [0, 0.2, 0.4], each of which has three levels. Here 0 means there should be no stations to be deployed or removed. Therefore, the first scale [0, 0.05, 0.1] means that at each timestamp there could be no, 5% or 10% of the total stations removed or deployed. Intuitively, this scale specifies how much our controller can explore in the search space. For instance, if the highest scale [0, 0.2, 0.4] were selected, in the extreme cases, the controller can add 40% of new stations, while removing 40% of the existing ones in one timestamp. Fig. 4.5d shows the improved `SC` and `NV` of different variants of our controller compared with the baseline **HD**. We see that as we increase the scale, i.e., our controller has more room to explore, the `NV` is relatively stable, but drops a bit as the scale reaches the highest. This means our approach can robustly optimise towards the goal of `NV`, while at high scales the increased randomness in generated deployment plans may deteriorate `NV`, e.g., deploying more stations may cost too much. On the other hand, for `SC` we see that it generally increases, which is also expected, since if more stations are allowed to be altered, the controller is likely to propose plans that quickly fill up the candidate stations to improve the service coverage.

4.6 Conclusion

In this chapter, we investigate the infrastructure optimisation problem for shared urban mobility systems, aiming to find the optimal deployment plan with which the system can achieve the desired service coverage, net revenue, and demand satisfied rate while being profitable. We design a high-fidelity simulation environment to capture the key operational details the shared mobility systems at fine granularity, and calibrate the environment with rich data collected from a real-world shared mobility system over 12 months. To tackle the deployment optimisation problem, we propose a novel multi-agent deep neural search algorithm, which employs hierarchical controllers to generate possible deployment plans. The controllers

are trained using a multi-simulation paradigm within our simulation environment, which learns to propose better plans in the next iteration. The proposed approach has been evaluated extensively, and experimental results show that: i) our neural search algorithm significantly outperforms both the baseline and state-of-the-art optimisation approaches, in various metrics including service coverage and net revenue; ii) By adjusting the weight between objectives, our search algorithm can optimise towards different directions, providing desired deployment plan for different cases; iii) the proposed hierarchical controller architecture reduces the search cost, where tuning the action space of the high-level controller has substantial impact on balancing exploration and exploitation. There are some limitations of the proposed approach. For example, it takes long time to train the model. Overall, the proposed approach achieved the aim to improve the shared mobility systems, and the next chapter studies how to improve such systems by managing the fleet in the systems.

Chapter 5

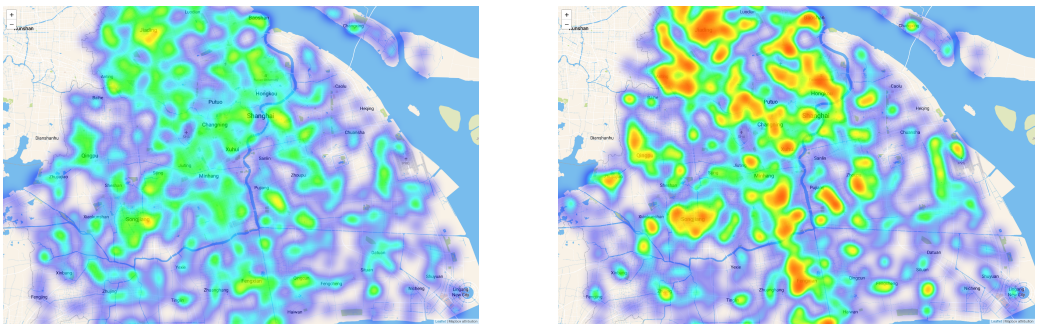
Fleet Management

5.1 Introduction

Fleets are indispensable components of the urban mobility systems. In this chapter, we investigate fleet management, in particular *rebalancing*, a key problem for the shared mobility systems to operate smoothly in urban settings. Despite their rapid adoption and growth across the world, almost all types of shared mobility systems face the problem of imbalanced distribution of their fleets as they operate over time. As in previous two chapters, in this chapter we also consider the shared e-mobility systems as our case study, which in fact are more challenging than standard shared mobility systems in this context, due to the extra constraints imposed by the use of electric vehicles (EVs). As stated before, the approaches developed in this chapter can be extended to other station-based shared mobility systems, by removing the EV related constraints.

To demonstrate the problem of imbalanced fleets, Fig. 5.1b shows the distribution of the EVs in a real-world shared e-mobility system at the last day of a month, where at the beginning of the month the system has just been manually rebalanced (see Fig. 5.1a). We see that after 30 days the vehicle distribution becomes very skewed, where some areas (red patches) have substantially more EVs than the others. In addition, such imbalance also happens within shorter time frames, e.g., we observe that in morning rush hours a large volume of EVs tend to flow to central areas and stay there, making fewer or even no vehicle available in other places. This would certainly have negative impact on the overall system performance, as potential customers may refrain from using the system if there is no available EVs nearby, or no parking location available near their destinations.

In fact, this rebalancing problem is a common issue in different types of



(a) 8am, Day 1 (b) 8am, Day 31

Figure 5.1: Vehicle distribution of a shared e-mobility system over a month, where hotter areas contain more vehicles. Images better viewed in colour.

shared mobility systems, e.g., shared bikes [153, 110, 154], taxi [12, 13, 51], and ride-sharing services [155, 156]. Typically, there are mainly two types of rebalancing strategies: i) employing a dedicated team of staff to manually reposition the vehicles/bikes, e.g., the work in [110]; or ii) incentivizing the users or drivers to voluntarily rent/return vehicles/bikes to the desired locations [157, 40]. In this chapter, we consider user incentive based rebalancing, since it is natural in our context to offer monetary incentives, e.g., price discount to the users, in exchange for them to reposition the vehicles to alternative destinations. However, unlike the traditional vehicle sharing systems, the rebalancing problem in shared e-mobility systems has two unique challenges. First of all, EVs typically have limited range, and the charging time is much longer than filling up the petrol or diesel vehicles. This adds many implicit constraints to the rebalancing problem, e.g., the EVs can't be repositioned to locations that is beyond the remaining range, and they also need to be sufficiently charged to serve future user orders. Secondly, as the shared e-mobility systems are relatively new to many cities, at this stage they tend to expand their services and infrastructure very rapidly. For instance, in the shared e-mobility system studied in this thesis, we observe that within just 12 months the total number of stations in operation has doubled, and in each month there are hundreds of new stations being deployed (see Fig. 5.2a). This makes the rebalancing task even more challenging, as at each timestamp the candidate stations to which the EVs may be repositioned are dynamically changing.

To address the rebalancing problem in the fast expanding shared e-mobility systems, we propose a novel multi-agent reinforcement learning (MARL) approach, which models the rebalancing task as a stochastic game among multiple agents. Each agent manages the EV stations within a spatial region, and learns to make

informed decision on where to reposition the incoming EVs to achieve maximum expected reward. To tackle the challenges of limited EV range and the charging delays, we propose to incorporate the range and charging information directly in our MARL algorithm, so that the agents are fully aware of those constrains when making decisions. For the dynamically expanding station networks, we propose a novel action cascading approach, which decomposes the actions of repositioning an EV into two subsequent and conditionally dependent sub-actions. The intuition is that when an EV needs to be repositioned, one could firstly decide which of the regions it should be redirected to, and then given the decision subsequently determine which station within that region should be the new destination. Therefore, the expansion dynamics are localised within the regions, while the first sub-actions should have static action spaces. In light of this, the proposed action cascading approach uses two connected policy networks to generate the sub-actions in sequel, where the second network is specially designed to handle the non-stationary action spaces.

There is also a solid body of existing work that uses the MARL formulation in rebalancing the shared mobility systems. For instance, the recent work in [110] uses a spatial-temporal DQN to rebalance the shared bikes, but it is fundamentally different from our work since it doesn't consider the dynamic system expansion. On the other hand, the work in [13] and [12] tackles the order dispatching problem in taxi systems, which although different from our problem, share similar challenges in the varying action spaces. However, their solutions are to allow the agents directly ranking the potential actions and selecting the one with highest score, while we use two policy networks to generate cascading actions and handle the non-stationarity. Concretely, the technical contributions are as follows:

- To the best of our knowledge, we are the first to identify the problem of rebalancing the continuously expanding shared e-mobility systems. We formulate the incentive-based rebalancing problem under the multi-agent reinforcement learning framework, and design the agents, states and rewards for the EV context accordingly.

- We conduct an in-depth case study of a real-world shared e-mobility system for one year, and collected rich sets of data from various sources to support our study. We analyse its operation model, expansion process and usage patterns. Based on the learned insights we consider a high-fidelity simulator for training and evaluation of the proposed approach.

- We propose a novel approach of policy optimisation with action cascading (ac-PPO), which uses two connected policy networks to handle the dynamics

introduced by the continuous expansion of the shared e-mobility systems. We also design a regularised reward function for the proposed ac-PPO, which can effectively stabilize training and improve data efficiency.

- The proposed approach has been evaluated extensively with our simulator, and the results show that our approach significantly outperforms the state-of-the-art, offering up to 12% improvement in net revenue and 14% in demand satisfied rate.

The rest of this chapter is orgnised as follows. We first discuss additional related work in Section 5.2 and briefly revisit the settings of the shared e-mobility systems, and discuss our key findings from the collected data that are relevant to the reblancing task in Section 5.3. Then we present the proposed approach for the rebalancing problem in Section 5.4. Section 5.5 evaluates the performance of our approach. We conclude the chapter in Section 5.6.

5.2 Related Work

As in the previous two chapters, in this section we discuss existing work in areas that are directly relevant to contributions in this chapter. For general background we refer the readers to Chapter 2.

5.2.1 Fleet Management

Existing work to address the problem of managing the fleet in the urban mobility services can be broadly categorized into three types, static reposition, dynamic reposition and user-based reposition. The first two are conducted by the system operators while the last is conducted by users. The static reposition solution is usually performed when the system is not operating or during the nights, with no perturbation from the users. Then the rebalancing tasks is cast into an optimisation problem with some objectives [37, 158, 159], e.g., maximising satisfaction of the customers. In practice, such static repositioning approaches only work well if the demand is predictable and stable. However, they can't perform rebalancing online as during operation the distributions of vehicles are varying. Dynamic reposition approaches consider the real-time flow of vehicles in the system, which also use optimisation techniques [157, 110, 160] to find the optimal repositioning plans. However, they depend heavily on the accuracy of demand prediction and it is often difficult to adjust the reposition operation given the unpredictable fluctuations in demand. The user-based reposition approaches solve the problem by incentivizing

the users with rewards to rent or return vehicles at specific stations [157, 40, 161]. However, it is often challenging to determine the optimal alternative station and estimate the appropriate reward to offer. Our work in Chapter. 5 falls into the last category, but unlike the existing solutions which assumes the system is static, we aim to tackle the rebalancing problem in the presence of dynamically changing station networks. It is fundamentally different from the static case, as at different time the candidate stations for potential repositioning operations can be different, which can't be addressed by the existing approaches.

5.2.2 Deep MARL for Mobility

Deep learning techniques have been used in various ubiquitous mobility applications due to their superior performance, e.g., improving service levels of taxi [51], predicting user demand [114], and reposition bikes or vehicles [40]. In particular, deep reinforcement learning has been introduced to solve various challenging mobility problems, such as traffic management [51, 162], order dispatching [13, 12], and rebalancing [110, 40]. Due to their distributed nature, many of those mobility applications can be modeled as multi-agent games, which can be well solved by deep reinforcement learning. For instance, the work in [110] designs a spatio-temporal reinforcement learning approach to dynamically reposition bikes in the bike-sharing system. The work in [12] proposes a multi-agent reinforcement learning framework to tackle the fleet management problem, while [13] addresses the order dispatching problem for ride sharing systems using mean field approximation. It has been shown that in those applications, deep reinforcement learning often achieves better performance, e.g., in terms of reducing potential customer loss, or increasing the gross merchandise values, than the traditional rule-based or optimisation based approaches, especially when the problem structure is complex. In this work, we also model the rebalancing problem in shared e-mobility system with MARL framework. However, our work differs from the existing work in that a) we extend the existing framework to directly model unique properties of EV sharing such as range limitations and charging time, and more importantly b) we develop the new action cascading technique to support continuous system expansion.

5.3 Rebalancing Shared e-Mobility Systems

In this section, we first remind the readers with some key concepts and assumptions of the shared e-mobility system considered in this thesis, and illustrate how it is operated in practice, highlighting the unique properties that are different from the

traditional mobility sharing services. Then we provide an in-depth analysis of the EV sharing data, particularly on the aspects related to fleet management and usage, and explain the problem of incentive-based EV rebalancing in the presence of continuous system expansion.

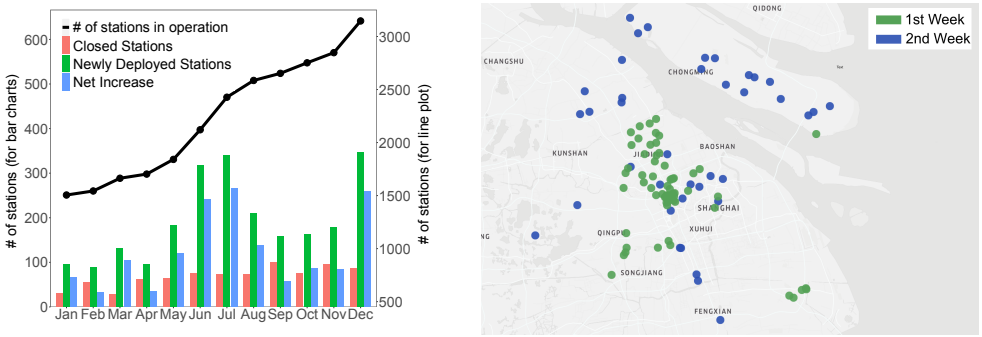
5.3.1 Shared e-mobility Systems

We follow the same models and assumptions for shared e-mobility systems as introduced in the previous two chapters (in Section 3.3.1 and 4.4.1). We assume that the electric vehicles (EVs) used in our shared e-mobility system are of limited range, and fully charged ranges of the EVs are fixed, and during normal driving the remaining range can be determined by a typical discharging model [150]. In addition, the time for charging of the EVs is much longer than refilling the traditional vehicles, which however can be estimated by a charging model [150], given the remaining range, battery capacities and charger specifications. We assume our system only uses one type of the chargers throughout. In Section 5.5 we will show how different EV ranges and charging duration may impact the patterns of system operation in more detail.

The stations S in the systems are the same as defined in Section 3.3.1, which also contain charging docks. The operation model of the shared e-mobility system is the same as in Section 4.4.1, where a user may pick up a vehicles from a certain station $s^{\mathrm{o}} \in S$, and return to the destination station $s^{\mathrm{d}} \in S$. As highlighted throughout this thesis, we assume the shared e-mobility system is continuously evolving during its operation, i.e., there are new EV stations deployed while existing ones closed at arbitrary time. In practice, a new EV station could be deployed in a new area to extend the coverage of current station network, or within the already covered areas to increase the station density. On the other hand, stations can also be closed temporarily or permanently for various reasons, e.g., with limited profit or the parking spaces are no longer available. We assume that overall the system keeps expanding, i.e., there are always more stations being deployed than closed.

5.3.2 Data Analysis for Rebalancing Task

Now we revisit the data collected for our study, and highlight the properties and characteristics that could affect the rebalancing task studied in this chapter. As motioned in previous chapters, we worked with a major shared e-mobility provider in Shanghai and collected its operational data for one year. Details of the data format and volume has been presented in Section 4.4.1. In particular, the data contains i) the complete *order transactions* of the shared e-mobility system, and



(a) (b)

Figure 5.2: (a) Statistics of system expansion during the 12 months period (taken from Fig. 3.1). (b) Newly deployed stations in two consecutive weeks.

ii) records on *station deployment*, i.e., when and where a station was deployed or closed. We refer the readers to Section 4.4.1 for more details.

Station Distribution. As discussed in the previous chapter, the density of the deployed stations is very different across the space. For instance, we find that the density decrease significantly towards the outer ring road, e.g., from 3.23 stations per km^2 in the inner ring road, to only 0.93 stations per km^2 in areas beyond the outer ring road. This would certainly affect the rebalancing decisions as in denser areas there may have more choices for alternative destinations, which are also closer. We also see that across the city the sizes of stations vary, and there are some particularly larger stations exist. By cross-checking the nearby POIs we found that those stations are typically within transportation hubs such as airports, which is also vital for rebalancing, as in those station there should be constantly significant volume of in/our demand, which needs sufficient amount of EV available to satisfy. We refer the readers to Fig. 4.3 for visualisation of the station and POI distributions.

System Expansion. Fig. 5.2a visualises the expansion of the shared e-mobility system studied in this work. We see that in 12 months time, the stations in operation has doubled from roughly 1500 to more than 3000, where in each month there are continuously hundreds of stations being deployed or closed. This would pose significant challenges to our rebalancing task, as the possible destinations for repositioning the EVs are continuously changing. Another observation is that in reality, the expansion process of the stations is not uniform, which requires the rebalancing strategies to adapt accordingly. For instance, Fig. 5.2b shows the newly deployed stations of the service in two consecutive weeks. We see that during the first week more stations were deployed at the central areas with only a few scattered around,

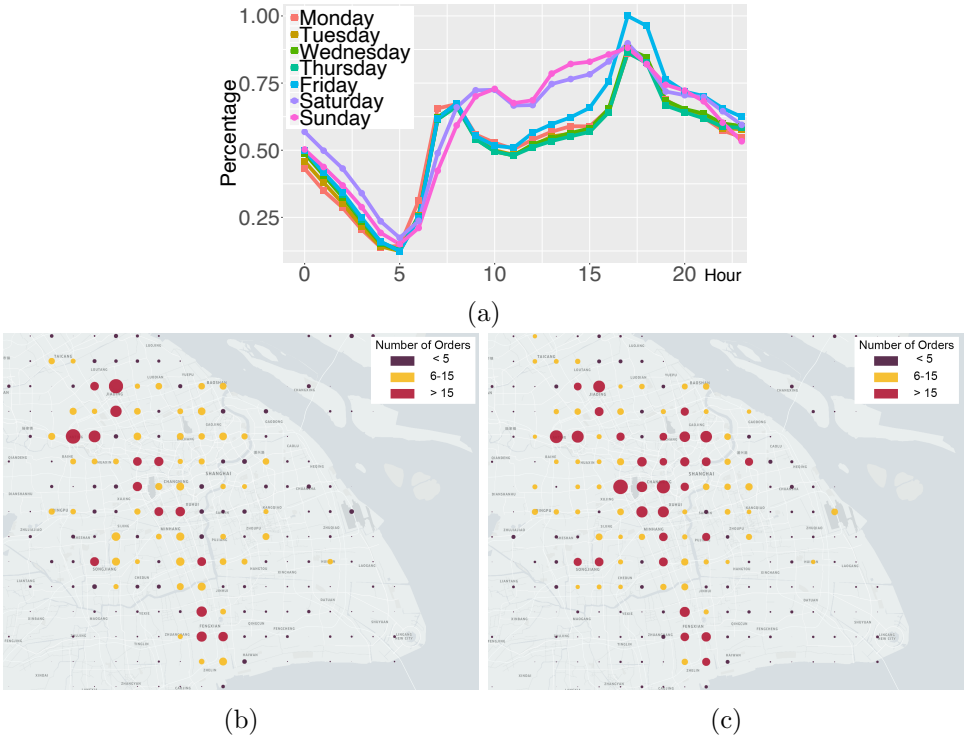


Figure 5.3: Demand patterns of the EV sharing service are highly imbalanced across space and time. (a) Average number of orders during each day in a week. (b) Average number of orders during morning rush hours. (c) Average number of orders during evening rush hours.

while in the second week stations were spread more uniformly. This indicates that as the system expands over time, different regions across the city might require different rebalancing strategies to better exploit the infrastructure available.

Demand Patterns. We observe that the demand patterns of the shared e-mobility system vary significantly across space and time. Firstly, there are clearly different temporal patterns between weekdays and weekends. As shown in Fig. 5.3a, we see the two peaks on weekdays align well with the morning (7-9am) and evening (5-7pm) rush hours in Shanghai. It is also interesting to see that the demand at evening peaks are higher than morning peaks. One possible reason is that people choose not to use EV sharing to avoid congestion during morning journeys to work, while are more flexible and willing to drive EVs when they finish in the evenings. Intuitively during those rush hours the rebalancing task is more challenging, where more rent/return demand surges across the city. For instance, we may have to reposition more EVs during those times, to ensure the balance of the system. In addition, Fig. 5.3b and Fig. 5.3c show the spatial distributions of demand at both

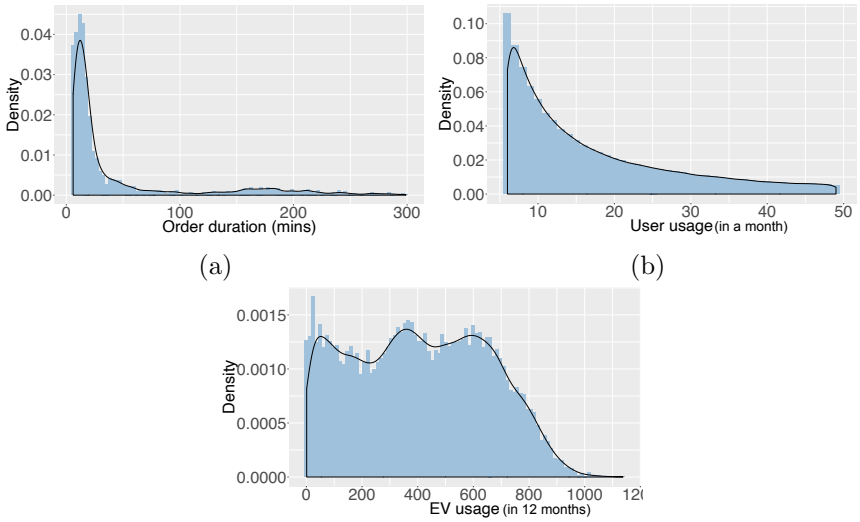


Figure 5.4: The distributions of (a) the order duration, (b) the monthly usage frequency per user, and (c) the overall EV utilisation.

mornings and evenings, where we see high volumes of demand are typically generated at suburban areas in the mornings, while at central areas in the evenings, reflecting the typical commuting needs. Therefore the rebalancing strategies should be able to take such spatio-temporal characteristics into account, and make informed decisions as how to reposition the EVs given the specific context.

Usage Characteristics. As shown in Fig. 5.4a, we see that on average the users of the shared e-mobility system tend to rent EVs for short trips, where the mean order length is 46 min. This indicates that for such systems, user-incentive based rebalancing is a sensible and beneficial choice, since users would not mind driving for a bit longer to reposition the EVs after shorter trips in exchange for incentives, while having more chances to reject such proposals after long journeys [157]. We also find that more than half of users tend to use the shared e-mobility system for less than 30 times in a month, i.e., roughly on daily basis, as shown in Fig. 5.4b. Therefore, although with appropriate incentives, the rebalancing strategies need to be carefully designed to not require excessive user efforts, e.g., repositioning EVs to far away stations. Interestingly, we also find the utilisation of EVs across the system is skewed. Fig. 5.4c shows the distribution of usage frequency of EVs (number of orders served) during the 12 month period. We see that some EVs have been used more than the others, e.g., they were deployed to popular stations such as the airports. This may lead to imbalanced vehicle conditions across the fleet, e.g., some EVs may suffer from early battery degradation or over wear and tear, which

should be avoided. In practice, rebalancing could also alleviate this, as EVs are often redistributed from popular stations to the quieter ones.

5.3.3 The EV Rebalancing Problem

In this chapter, we consider rebalancing the shared e-mobility system by incentivizing the users. In particular, let $o_t = (s^{\mathrm{o}}, s^{\mathrm{d}})$ be an order placed by a user at time t, requesting to rent a vehicle from station s^{o} and return to s^{d}. As discussed above, due to the very unbalanced user demand over space and time, during operation the shared e-mobility system can become skewed where some stations are too crowded (i.e., not enough places to park/charge), while the others are depleted (i.e., not enough EVs to rent). Clearly both cases would negatively impact the system performance in satisfying future user demand. Therefore, to alleviate such imbalance we may have to reposition the EV serving the current order o_t to another station $s^{\mathrm{d}'}$ instead of the original destination s^{d}, if the remaining range of the EV is sufficient. We motivate the user who is driving the EV to perform this for us by offering her a certain amount of monetary reward. In general, we offer her a reward of value $v(s^{\mathrm{d}}, s^{\mathrm{d}'})$ which depends on the extra distance she has to drive from s^{d} to the new destination $s^{\mathrm{d}'}$. The user may or may not accept the offer, according to a prior user model [157]. If she accepts, we pay the reward directly e.g., discounting the order price, while otherwise we allow the user to return the EV to her original destination and charge the order normally.

Therefore, the rebalancing problem studied in this chapter is that given the total available budget B on user incentives, for each order $o_t = (s^{\mathrm{o}}, s^{\mathrm{d}})$, we want to decide where to reposition the EV to minimise the future customer loss (i.e., satisfying as much user demand as possible) while maximizing the net revenue of the shared e-mobility system, in the presence of limited EV range, typical EV charging time, and the dynamically expanding station network.

5.4 Rebalancing Methodology

To tackle this EV rebalancing problem, in this section we first formulate it as a Multi-Agent Reinforcement Learning (MARL) task with non-stationary action spaces in Section 5.4.1. Then in Section 5.4.2 we explain in the stationary cases how MARL tasks can be solved by general policy optimisation techniques, and finally present the proposed policy optimisation approach with action cascading in Section 5.4.3, which is able to handle the non-stationarity in our problem.

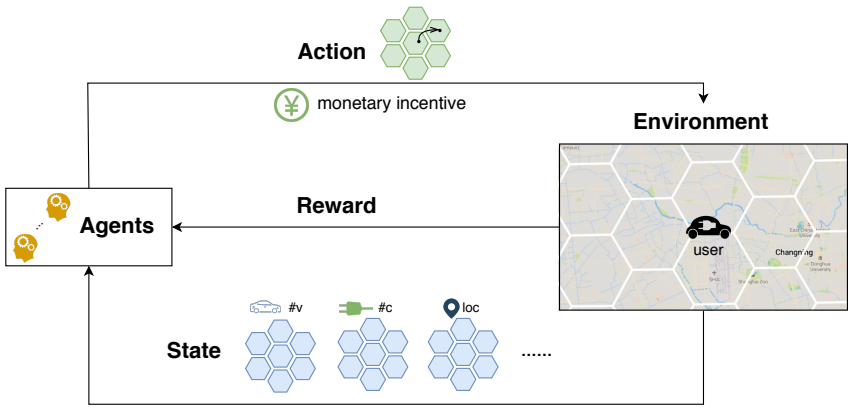


Figure 5.5: An illustration of the MARL formulation for the fleet rebalancing task.

5.4.1 Fleet Rebalancing as a MARL Task

In this problem, we consider a set of autonomous agents, each of which has its own action space, and interacts with the common environment to improve its behavior. Comparing to the single agent settings, the multi-agent formulation is more suitable to address our problem, because multiple agents could better exploit the decentralized nature of the rebalancing task, and by design it allows new agents to join online in a flexible way, which could capture the dynamic system expansion in our context. In addition, under the MARL framework learning can be more efficient than the single agent case, as the action spaces of the agents are much smaller, and the computation can be accelerated by parallel processing. Formally, we model the above rebalancing problem in shared e-mobility systems as a Markov Game $G = (N, \mathcal{X}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where at most N agents interact with the environment, characterized by the states $\mathcal{X}$ and transition functions $\mathcal{T}$. $\mathcal{A}$ is the joint actions of the agents, $\mathcal{R}$ is the reward functions, and γ is the discount factor. Note that the MARL formulation for the rebalancing task bears some resemblance to that of the infrastructure optimisation problem discuss in Chapter 4, but they also differ in many aspects, from the design of action space and reward function, to implementation details. In the following we present the specific MARL formulation for the rebalancing task studied in this chapter.

Agents. In our formulation, we assume an agent controls the operation of EV stations within a certain geospatial region. Essentially, we consider the agent as the virtual system operator of that region, who decides if an EV returning to stations within its region should be repositioned and where. Without loss of generality, in this chapter we assume each agent manages a hexagonal region, i.e., the space is partitioned into hexagonal grids with the same size. As the shared e-mobility system

is continuously expanding to new areas, in our case the number of agents at a given time t is a variable N_t, but we assume the maximum number of agents are fixed N, which is determined by the maximum number of hexagonal grids in the space partition.

States. At time t, the global state $\boldsymbol{x}_t$ contains information about the the currently online EV stations, the available EVs and the spatial distributions of the user demand. More concretely, we assume $\boldsymbol{x}_t$ is the combination of states for each hexagonal grid $\boldsymbol{x}_t = \{x_t^i\}$, $i \in [1, N]$. For the i-th hexagonal grid, its state x_t^i encodes information about the stations within the boundary of this grid. In particular, for each station we consider its location #`loc`, number of available charging docks #`c`, number of EVs parked in the stations #`v` and their individual range, as well as the potential future rent/return requests (number of EVs) and the average value of potential future orders in the next timestamp.

Agent Observations. Let agent i manage the i-th hexagonal grid. At time t, we assume the agent makes a partial observation of the global state $\boldsymbol{x}_t$. In this work, we assume that an agent can draw observations from only the grids within its one and two-hop neighborhood, i.e., normally it observes the states of itself and the 19 grids around it. This means the receptive fields of our agents are two-hop, which enables them to observe and interact with the local environment around them, and learn to better cooperate with their neighboring agents.

Actions. For agent i, its action a_t^i describes how each EV returned to the grid i at time t will be handled, i.e., continue directly to the original destination station, or being redirected to another station. In this work we assume that our agents shall only propose to reposition the EVs to stations within the one-hop neighborhood of destination grid to avoid excessive user effort, i.e., an EV being returned to grid i shall only be repositioned to stations in the 6 neighbor grids of i if necessary. As there are stations being deployed or closed dynamically in our system, the action space $\mathcal{A}_t^i$ of the agent i is non-stationary, i.e., the candidates of the reposition stations may vary over time. Therefore at time t, a joint action $\boldsymbol{a}_t \in \mathcal{A}_t^1 \times ... \times \mathcal{A}_t^{N_t}$ specifies the EV reposition strategies of all N_t agents.

State Transitions. The state transitions $\mathcal{T}$ are defined as $\mathcal{T}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{a}_t, \boldsymbol{u}_t)$, where $\boldsymbol{x}_t$ is the previous state, $\boldsymbol{a}_t$ is the joint action of the agents, and $\boldsymbol{u}_t$ is the control input describing the system dynamics. We use this variable $\boldsymbol{u}_t$ to capture the changes caused by the station network expansion happened at t, i.e., which new stations are deployed with how many new EVs, and which exiting stations are off-line from time t. Therefore, in our case the environment is non-stationary, as the state transitions are induced not only by the actions $\boldsymbol{a}_t$ of the agents, but also the external input $\boldsymbol{u}_t$,

which is often governed by a random process.

Reward Function. For each agent, the reward r_t^i of taking an action at time t is determined by the reward function $\mathcal{R}^i(\boldsymbol{x}_t, \boldsymbol{a}_t)$, as defined in Eq. (4.2). As discussed above, in the rebalancing task we would like to maximise the gross revenue of our shared e-mobility system with minimum cost on user incentives. Intuitively, the gross revenue can be increased by providing more EVs to stations with higher expected order values. However, in practice we found that considering only the order values would lead to over-greedy agents, who would only push EVs to certain "hot" stations with higher average order values such as the airport but ignore the others, causing further imbalance to the system. Therefore, to mitigate that we also reward the agents that choose to reposition the EVs to the stations that are in shortage of vehicles, i.e., those are likely to have more orders in the future but currently don't have enough EVs available. Concretely, suppose that at time t the agent i decides to reposition an incoming EV to a new destination station $s^{\mathrm{d}'}$ instead of the original s^{d}. To balance fairness and the potential revenue, we design the reward function r_t^i as follows:

$$r_t^i = g_t^{\mathrm{d}'} + \alpha_1 v_t^{\mathrm{d}'} + \alpha_2 b_t^{\mathrm{d}'} - \alpha_3 d(s^{\mathrm{d}'}, s^{\mathrm{d}}) \tag{5.1}$$

where $g_t^{\mathrm{d}'}$ is the expected demand gap at station $s^{\mathrm{d}'}$ in the next timestamp, i.e., the number of orders minus the number of available EVs onsite, and $v_t^{\mathrm{d}'}$ is the expected average order value at station $s^{\mathrm{d}'}$. $b_t^{\mathrm{d}'}$ is a binary variable indicating if station $s^{\mathrm{d}'}$ is empty, i.e., there is no EVs in the station at t. We use $b_t^{\mathrm{d}'}$ to explicitly encourage agents to position EVs to those empty stations, which are very likely to cause unsatisfied demand in the future. The penalty term $d(s^{\mathrm{d}'}, s^{\mathrm{d}})$ is the cost we pay (monetary reward to the user) for this reposition, which is proportional to the squared distance that one has to travel from the original s^{d} to the new destination station $s^{\mathrm{d}'}$ [40]. The weights α_1, α_2 and α_3 scale the different reward/penalty terms to approximately the same range, which are set empirically in our experiments via grid search. Given the reward function, each agent aims to maximise its discounted reward $\mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k}^i]$, where $\gamma \in [0, 1]$ is the discount factor. Fig. 5.5 shows an illustration of the MARL formulation of the rebalancing task.

It is worth pointing out that in our MARL formulation, the action spaces of the agents are non-stationary, due to the fact that the EV station network is dynamically evolving over time. This means in both training and testing, at different timestamps an agent may face different lists of candidate stations, e.g., at arbitrary locations and with heterogeneous properties such as number of charging docks or available vehicles, to which it may reposition the EVs. In that sense, typical RL techniques such as Q-learning or policy-based approaches are not directly applicable,

as they often require a fixed set of actions to evaluate their Q values or the probability distribution over them. In the following, for completeness we first briefly introduce the standard policy optimisation approaches for multi-agent RL that are designed for stationary action spaces, and then explain how we extend them to deal with the non-stationary cases.

5.4.2 Standard Policy Optimisation

Depending on specific learning tasks, there are two main types of model-free MARL algorithms in practice: the value-based and policy-based approaches. The former relies on good estimations of the state action value functions such as DQN [163], but often fails when the rewards are complex. On the other hand, the policy-based approaches directly search for better policies in each iteration, but naive techniques such as policy gradient [164] suffer from large gradient variances and are not robust. In this work, we consider the more recent policy optimisation algorithms [165], which combine the best of two worlds: the learning task is cast into a constrained optimisation problem, which iteratively searches for better policies that yield larger returns.

More concretely, let π_θ be the policy parameterized by θ, which is often implemented as policy networks. In our case, the agents aim to maximise the expected discounted reward since the beginning of time: $\eta(\pi_\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}_t\right]$, where $\boldsymbol{r}_t$ is the joint reward. In practice, $\eta(\pi_\theta)$ can be optimized by the Minorize-Maximization (MM) algorithm [166], which tries to find a surrogate function approximating the lower bound of η at the current policy π_θ and optimize it iteratively. In particular, we consider the following objective function L:

$$L(\theta) = \hat{\mathbb{E}}\left[\frac{\pi_\theta(\boldsymbol{a}_t|\boldsymbol{x}_t)}{\pi_{\theta_{\text{old}}}(\boldsymbol{a}_t|\boldsymbol{x}_t)}\hat{A}_t\right] \tag{5.2}$$

where $\boldsymbol{a}_t$ and $\boldsymbol{x}_t$ are the joint actions and states at time t, and $\hat{\mathbb{E}}[\cdot]$ is the empirical average over the batch of samples. $\hat{A}_t$ is an estimator of the advantage function A_t, which is the benefit of taking a specific action $\boldsymbol{a}_t$ under the current state $\boldsymbol{x}_t$ than the expected state value:

$$A_t(\boldsymbol{a}_t, \boldsymbol{x}_t) = Q(\boldsymbol{a}_t, \boldsymbol{x}_t) - V(\boldsymbol{x}_t) \tag{5.3}$$

where $Q(\boldsymbol{a}_t, \boldsymbol{x}_t)$ is the Q-function and $V(\boldsymbol{x}_t)$ is the value function.

Here the actions $\boldsymbol{a}_t$ are drawn from the current policy $\boldsymbol{a}_t \sim \pi_\theta(\boldsymbol{a}_t|\boldsymbol{x}_t)$, and we assume the states transition only depends on the previous states and actions:

$\boldsymbol{x}_{t+1} \sim \mathcal{T}(\boldsymbol{x}_{t+1}|\boldsymbol{a}_t, \boldsymbol{x}_t)$. Note that we use the advantage function A_t instead of the Q function which is the expected reward, to reduce the estimation variance and stabilize training. The intuition is that L approximates $\eta(\pi_\theta)$ locally at the current policy π_θ, but can get inaccurate as it moves away from π_θ. Therefore, to avoid updating the policy too much, we consider the following constrained optimisation problem:

$$\underset{\theta}{\text{maximise}} \quad L(\theta) - \lambda \hat{\mathbb{E}}[D_{\text{KL}}(\pi_\theta(\cdot|\boldsymbol{x}_t), \pi_{\theta_{\text{old}}}(\cdot|\boldsymbol{x}_t))] \tag{5.4}$$

The constraint term is the mean KL divergence between the old and current policy, indicating that the new policy can't be too different from the old one. We then iteratively optimize the objective function, until the optimal policy π_{θ^*} can be found. Note that the standard policy optimisation algorithms assume that the action spaces of the agents are stationary, i.e., at each timestamp the learned policy π_θ outputs a distribution over the fixed set of possible actions $\mathcal{A}$. In the context of our rebalancing task, this means the standard algorithms can only work if the station network is static, i.e., the candidate stations to which an agent can reposition EVs should be fixed, and thus they are not directly applicable to our problem where the station network is dynamically changing.

5.4.3 Policy Optimisation with Action Cascading

We now present the proposed policy optimisation approach with action cascading (ac-PPO), which extends the standard algorithms and is able to handle the non-stationarity in our EV rebalancing problem. The key intuition is that in our settings the action of repositioning an EV to an alternative station can be viewed as a sequence of two sub-actions, where we first decide which grid the EV should go to, and then figure out which station within that selected grid should be the target. In essence, we chain two sub-actions, one inter-grid and the other intra-grid, to achieve the desired goal of repositioning the vehicle. One of the benefits of this action cascading is that now the inter-grid actions can have fixed action spaces, while the non-stationarity of the station network would only affect the intra-grid actions. This makes it possible to fit our problem into the policy optimisation framework discussed above, where the non-stationarity within different grids can be handled by separate policy selectors. In the following, we first explain the design of action cascading in more detail, and then we show how we adapt the reward structure to stabilize training.

Action Cascading. Let a_t^i be the action of agent i at time t. We assume a_t^i can be decomposed as $a_t^i = (ga_t^i, sa_t^i)$, where ga_t^i is the inter-grid action that decides

which grid within the neighborhood the EV should be redirected to, and sa_t^i is the intra-grid action which determines the actual destination station within the selected grid. Clearly, here ga_t^i has a fixed action space, which contains the six neighbors around the gird i and itself. Therefore, ga_t^i can be sampled from the output of a standard policy network π_θ^g as discussed in Section 5.4.2. Let us assume that we have a ga_t^i that would redirect the EV to a nearby grid j. Now we need to find the intra-grid action sa_t^i that selects a suitable station within grid j. Note that here the action space of sa_t^i is not stationary, as there are always stations deployed or closed in grid j. We address this by using a special action-in policy network π_ϕ^s as shown in Fig. 5.6, which takes the current state x_t^j of the grid j, and the output of the last layer of the inter-grid policy network π_θ^g as the input. Note that here the state x_t^j encodes information of the current stations and vehicles within grid j, and the output from π_θ^g conditions x_t^j. The output of the network π_ϕ^s are deterministic values of each station within grid j, and we select the one with highest value for action sa_t^i.

Essentially, we use two policy networks that are connected, to determine the inter-grid and intra-grid actions respectively. During training, we only sample from the inter-grid policy network π_θ^g, while considering the intra-grid policies of π_ϕ^s are deterministic, which makes the training more data efficient. In our implementation, we train the networks with the following clipped objective function:

$$L^{\text{CLIP}}(\theta, \phi) = \hat{\mathbb{E}}\left[\min(R_t^\theta \hat{A}_t^{\theta,\phi}, \text{Clip}(R_t^\theta, 1-\epsilon, 1+\epsilon)\hat{A}_t^{\theta,\phi}\right] \tag{5.5}$$

where R_t^θ is the probability ratio between new and old inter-grid policy:

$$R_t^\theta = \frac{\pi_\theta^g(\boldsymbol{ga}_t|\boldsymbol{x}_t)}{\pi_{\theta_{\text{old}}}^g(\boldsymbol{ga}_t|\boldsymbol{x}_t)} \tag{5.6}$$

This R_t^θ together with the Clip function constrains the policy updates to avoid obtaining very different new policies. ϵ is the hyperparameter, which is usually set to values around 0.2$\sim$0.3. Note here we only consider the inter-grid policy π_θ^g, since the output $\boldsymbol{ga}_t$ has stationary action space. On the other hand, the advantage function $\hat{A}_t^{\theta,\phi} = Q(\boldsymbol{a}_t, \boldsymbol{x}_t) - V(\boldsymbol{x}_t)$ considers both inter and intra-grid policies, since the reward $\boldsymbol{r}_t$ is given to the full actions $\boldsymbol{a}_t = (\boldsymbol{ga}_t, \boldsymbol{sa}_t)$, where the Q function is evaluated with the discounted rewards of the actions obtained in this experience.

Reward Regularisation. Essentially, the proposed ac-PPO addresses the non-stationarity in action spaces by decomposing the action into the sequence of inter-grid and intra-grid sub-actions, and using two connected policy networks to determine them. Therefore, we fit the non-stationary rebalancing problem into the policy

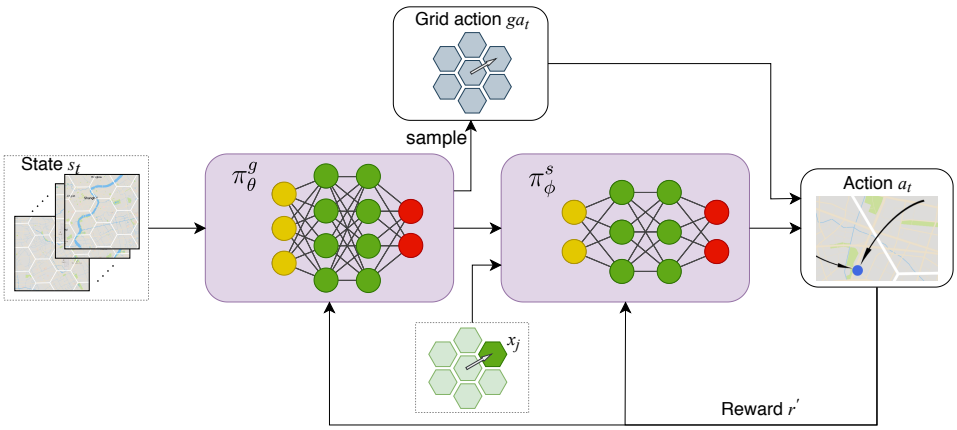


Figure 5.6: Overview of the proposed action cascading.

optimisation framework, by allowing non-stationary reward functions. In fact, from the view of the inter-grid policy network π_θ^g, the reward distribution of the same action (e.g., repositioning the EV to the grid directly above) across different timestamps may be different, because the set of stations within the grid are changing, and the intra-grid policy network π_ϕ^s is very likely to select different destination stations. When the station network is very dynamic, such non-stationarity in reward could lead to large gradient variance when training π_θ^g. To address that, we propose to regularise the reward function r_t^i in Eq. (5.1) with a baseline:

$$r_t^{i'} = r_t^i + \beta \bar{r}_t(j) \tag{5.7}$$

where the regularisation term $\bar{r}_t(j) = \bar{v}_t(j) \cdot \bar{g}_t(j)$ is the product of the mean order value $\bar{v}_t(j)$ and the average future demand gap $\bar{g}_t(j)$ (# of user demand - # of available EVs) per station in grid j, assuming that the action is to reposition the EV to a station in the target grid j. Intuitively, $\bar{r}_t(j)$ can be viewed as the "potential" of the grid, indicating how much extra revenue one would expect to get if more EVs are repositioned to stations within the grid. In practice, $\bar{r}_t(j)$ is updated every timestamp and is more stable than r_t^i, which depends on the particular destination station selected by the intra-grid policy network π_ϕ^s. The weight β scales the regularisation term to adjust its impact during learning. We describe the detailed ac-PPO algorithm in Alg. 1.

Algorithm 1 Policy Optimisation with Action Cascading

while training not finished **do**
  for agent $i = 1, 2, \ldots$ **do**
    for $t = 1, 2, \ldots, T$ **do**
      Run inter-grid policy $\pi^g_{\theta_{old}}$ to sample ga^i_t
      Given $ga^i_t = j$, run intra-grid policy $\pi^s_{\phi_{old}}(j)$ to get sa^i_t
      Take the cascading action $a^i_t = (ga^i_t, sa^i_t)$
      Collect reward $r^{i'}_t$ as in Eq. (5.7)
      Compute advantage estimates $\hat{A}_t$ as in Eq. (5.3)
    end for
    Sample a batch of experiences as described above
    Optimize the objective function $L^{\text{CLIP}}(\theta, \phi)$ as in Eq. (5.5)
    Update $\theta_{old} \leftarrow \theta$
    Update $\phi_{old} \leftarrow \phi$
  end for
end while

5.5 Evaluation

In this section, we evaluate the proposed MARL algorithm extensively with a custimised version of the simulation environment as presented in Chapter 4. We first explain our experimental settings and competing approaches in Section 5.5.2, and report the experiment results in Section 5.5.3.

5.5.1 Simulation Settings

For the rebalancing task, we consider the simulation environment designed in Section 4.4, and adapt it to EV model and simulate the rebalancing operations in shared mobility systems. As in the previous chapter, we consider 10 mins as one timestamp, i.e., one day (24 hours) contains 144 time intervals. The space is partitioned into hexagonal grids, where each agent manages one grid and can reposition EVs to the neighboring 6 grids. In total we have 598 grids covering the entire city of Shanghai. To simulate the dynamic system expansion, our simulator uses a random process to control i) the expansion speed, i.e., numbers of new stations to be deployed and existing stations to be closed at a timestamp; and ii) the expansion plan, i.e., where to deploy the new stations, and which existing stations should be closed. The parameters of the random process are learned from the real expansion data. For a newly deployed station, the simulator sets the number of parking/charging docks using the actual station data, i.e., it finds the set of stations in real data that are close to this new station in simulation, and use their average number of charging

docks to initialize it. We also allocate certain number of EVs to this new station, according to a probability ratio learned from the historical expansion data.

We use similar process as in Section 4.4 to generate the user demand, i.e., user orders for the shard mobility system. At initialisation, the simulation environment assumes all the EVs in the system are fully charged. As the simulation progresses, the simulator keeps track of the range of each EV by applying a discharging model [150] when serving orders. Once the order is completed, i.e., the EV is returned to a station and charging, the remaining range of the EV is estimated using a charging model [150].

When there is a need to reposition an EV, our simulation environment computes an offer of monetary reward, whose value depends on the square of the extra distance that the user has to travel [157]. The simulator assumes that the user would accept this offer according to an incentive acceptance probability p, which indicates how cooperative the user is. In our experiments, we vary p and study how different p values could impact the rebalancing performance. If the offer is accepted, the simulator updates the order information accordingly (e.g., discounting the price, changing the destination station), and also updates the status of the EV and stations accordingly. We calibrate the simulation environment with real-world data in the same way as discussed in Section 4.4.5.

5.5.2 Experimental Setup

In our experiments, we compare the proposed approach with the following baselines:

- **No Rebalancing (NR)**, which simulates the operation of shared e-mobility system without any rebalancing actions.
- **Random Rebalancing (RND)**, where in rebalancing the EVs are repositioned randomly to nearby stations (with at least one charging dock available) within a certain radius of the original destinations.
- **Revenue Greedy (REV)**, which is similar with RND but selects the stations with the highest average order values.
- **Demand Gap Greedy (DMD)**, which prefers the stations with the highest demand gap in the vicinity.
- **STRL**, which is our implementation of the the state-of-the-art rebalancing approach [110] for shared mobility systems. It uses multi-agent spatial-temporal reinforcement learning to reposition shared bikes across different stations.

To further validate the performance of the proposed action cascading, we also consider different variants of our MARL algorithm for ablation study. For the inter-grid actions (i.e., determining which grid for repositioning), we consider the following different implementations of the policy network π_θ^g (as explained in Section 5.4.3):

- **Policy Gradient (ac-PG)**, which uses the standard policy gradient technique to determine the inter-grid actions. The policy network is implemented with a four layer MLP, and we use learning rate 4e-4.
- **Deep Q Networks (ac-DQN)**, which uses a Q-network to approximate the action-state values. In our implementation, we use four layer MLP and ϵ-greedy policy as the agent policies, where ϵ is annealed from 0.1 to 0.02 in training. The learning rate is set to 5e-4.
- **Advantage Actor Critic (ac-A2C)**, which uses two separate networks (the actor and the critic) to produce actions and estimate the advantage values respectively. In our implementation, we use two four layer MLPs for the actor and critic network. We use learning rate 1e-4 for both networks.
- **Proximal Policy Optimisation (ac-PPO)**, which is the policy optimisation approach as discussed in Section 5.4.3. In particular, we use a four layer MLP as the policy network to generate the inter-grid actions, which also estimates the state values. The learning rate is set to 5e-5.

Note that for all the above variants, we use the proposed intra-grid policy network π_ϕ^s as described in Section 5.4.3 to determine the later intra-grid actions that redirect the EV to destination stations, and the same function in Eq. 5.7 to collect rewards. On the other hand, to evaluate the performance of different approaches for the intra-grid policy network π_ϕ^s, we fix the inter-grid policy network π_θ^g as the PPO, and consider the following additional variants:

- **PPO + Random (PPO+RND)**, which uses PPO to determine which grid to reposition the EV, and then randomly selects a destination station within that grid.
- **PPO + Revenue Greedy (PPO+REV)**, which is similar to the above, but instead of random selection, it finds the destination station with the highest average order values.

	NR	RND	REV	DMD	STRL	ac-DQN	ac-PG	ac-A2C	ac-PPO	PPO+RND	PPO+REV	PPO+DMD
DS	74.69%	49.79%	82.15%	81.09%	82.47%	83.50%	83.64%	85.23%	**88.79%**	53.94%	83.19%	82.88%
ΔDS	—	-24.90%	7.46%	6.41%	7.78%	8.81%	8.95%	10.55%	**14.10%**	-20.75%	8.51%	8.20%
ΔGMV	—	-36.30%	8.25%	3.22%	9.27%	10.76%	11.26%	13.13%	**18.13%**	-29.82%	10.41%	6.22%
ΔNV	—	-47.71%	-7.64%	-0.48%	1.12%	6.95%	7.37%	8.53%	**12.23%**	-48.40%	-3.27%	1.72%
$\Delta\|\boldsymbol{o}\|/\|\boldsymbol{a}\|$	–	—	9.28	4.98	2.08	1.14	1.09	1.11	1.12	—	7.82	3.53

Table 5.1: Performance of the competing approaches in 1) demand satisfied rate (DS), 2) increased demand satisfied rate (ΔDS) w.r.t. baseline NR, 3) increased % of GMV (ΔGMV) w.r.t. baseline NR, 4) increased % of net revenue value (ΔNV) w.r.t. baseline NR, and 5) # of increased order per reposition operation ($\Delta|\boldsymbol{o}|/|\boldsymbol{a}|$, only showing positive values).

- **PPO + Demand Gap Greedy (PPO+DMD)**, which decides the destination station by finding the one with the largest demand gap within the grid.

In our experiments, all the competing approaches are implemented with TensorFlow 1.14.0, and trained with a single NVIDIA 2080Ti GPU. For efficient training, in our implementation all the agents share the same policy and value networks, which also encourage them to collaborate with the others. In practice, the agents can maintain their own network parameters locally, and get updates from a central server. To be fair, we assume that for all approaches the amount of user incentives we have to pay for a particular reposition action is calculated by the same cost model [157], which depends on the squared distance between the original destination and proposed reposition station. We evaluate the competing approaches against two main metrics:

Demand Satisfied Rate (DS), which is the percentage of the demand satisfied by an algorithm with respect to the total user demand generated.

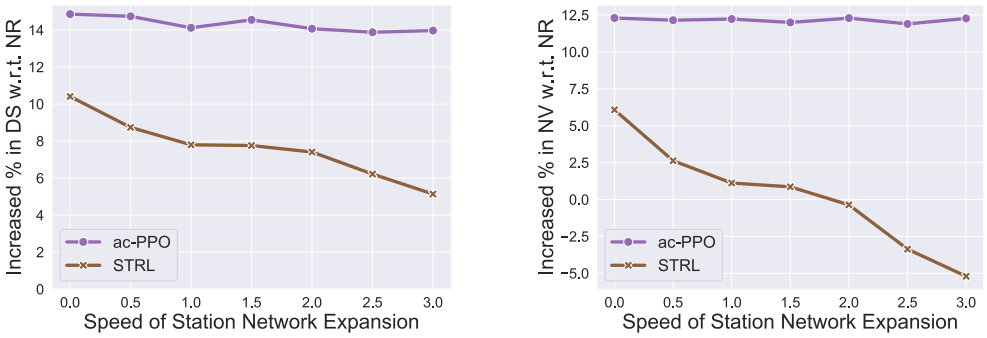
Net Revenue Value (NV), which is calculated as the GMV of the system subtracts the cost on user incentives.

5.5.3 Results

Overall Rebalancing Performance. The first set of experiments evaluate the overall rebalancing performance of different approaches. Table. 5.1 shows the demand satisfied rates and the increased net revenue (in percentage) of the competing algorithms. We allow the station network to expand at the normal speed (similar with the real data), where at each timestamp there are new stations deployed and existing ones closed. We can see that comparing to no rebalancing (NR) which is the normal operation, randomly selecting the reposition stations (RND) won't help in neither satisfying user demand, nor improving net revenue: we observe a signif-

icant drop in both performance metric. On the other hand, if we are greedy on order values (REV) we do satisfy more user demand by roughly 7%, but the net revenue actually drop by 8%. This is because with this algorithm, the agents tend to excessively reposition EVs to those station with high expected order values, while ignore the cost on user incentives. In our experiments, we find that on average, REV would satisfy one extra order (which tends to be of higher values) at the cost of repositioning 9.3 EVs. On the other hand, the demand gap greedy algorithm (DMD) achieves more balanced performance, improving the demand satisfied rate (DS) by 3%, while maintaining similar net revenue with the baseline NR. This is also expected, as sending EVs to stations with larger demand gap is more likely to fulfill future user orders. On average, this DMD has to reposition 4.9 EVs to satisfy one extra order, which is better than REV. We observe that the state-of-the-art STRL outperforms the baselines, with 8% improvement in DS and 1% improvement in net revenue (NV). It also has more efficient repositioning as well: on average it repositions 2.1 EVs to satisfy one extra order. It confirms that by using spatial-temporal RL, the STRL can better learn the demand pattern across space and time, and thus make more informed decisions in rebalancing. However, we see that the approaches with the proposed action cascading significantly outperforms STRL. For instance, the best ac-PPO can achieve almost 15% improvement in demand satisfied rate, while obtaining approximately 12% more net revenue. This means comparing to the state-of-the-art STRL, our approach can offer two-fold improvement in satisfying user demand, while >10x net revenue improvement. In addition we find that on average approach only needs to reposition 1.1 EVs to satisfy an extra order, which is very efficient. This validates the effectiveness of the proposed action cascading, and also shows that our approach can cope with the dynamically expanding station network. We will show later that the gap between our approaches and the STRL would be even larger when the system expansion dynamics increases.

Performance of Inter-grid Policy. This experiment compares the performance of different algorithms to learn the inter-grid policy π_θ^g in our action cascading framework, which decides the grid that the EVs should be repositioned to. We consider four different learning algorithms, the policy gradient (PG), DQN, A2C and PPO. Note that here we plug in those algorithms to our action cascading framework, while using the same intra-grid policy network π_ϕ^s later and feed the algorithms with the same reward. Firstly, we see that even the weakest performed algorithm ac-DQN can achieve better performance than the state-of-the-art STRL, with approximately 5% improvement in demand satisfied rate and 1% increase in net revenue. This is because STRL doesn't have the mechanism of handling station network expansion,



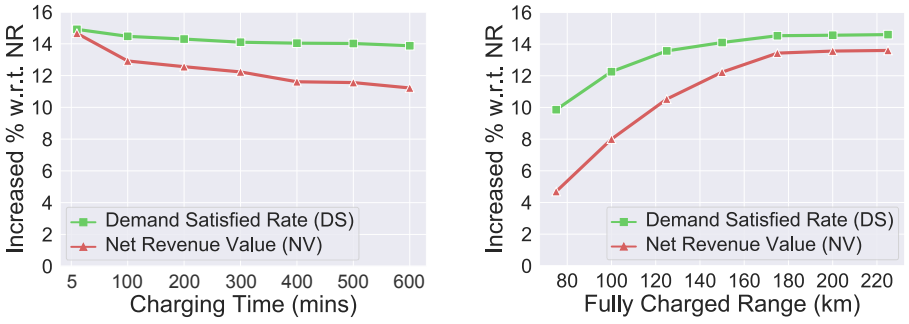
(a) Increased % in DS

(b) Increased % in NV

Figure 5.7: Performance of the proposed ac-PPO and STRL under different speeds of station network expansion.

while the proposed action cascading can effectively work with the non-stationarity in action space. The reason why ac-DQN performs not so well is because that DQN typically works well when the action space is finite with clear reward structures, while in our case although action cascading can address varying action spaces, the non-stationarity in rewards can still affect the performance of ac-DQN. On the other hand, policy gradient (PG) only performs slightly better than DQN, but is inferior to A2C, which is about 2% better in both satisfied demand rate and increase in net revenue. This is because in practice the variance of the gradients computed by PG can be large, and thus it is very likely to deviate from the optimal direction if the learning rate is too high. Comparing to ac-A2C, the ac-PPO (discussed in Section 5.4.3) provides a further improvement of roughly 4% in both demand satisfied rate and net revenue, achieving >14% better demand satisfied rate and >12% extra net revenue than no rebalancing (NR). This maps to more than 200,000 USD extra revenue per month according to the real data where the mean order value is around 3.8 USD and average number of orders per month is about 500k.

Performance of Intra-grid Policy: The third set of experiments studies the performance of different approaches in generating the Intra-grid policies. Here we use the best performing PPO algorithm to output the inter-grid actions. We compare the proposed ac-PPO with three variants, where we replace the intra-grid policy network π^s_ϕ with rule-based strategies: the random (PPO-RND), the revenue greedy (PPO-REV), and the demand gap greedy (PPO-DMD). Essentially, here we consider a vanilla version of action cascading, where inside the grids we follow certain heuristics to find the destination. For fair comparison, we use the same reward function as in our ac-PPO for all the algorithms. As shown in Table. 5.1, the random



(a) (b)

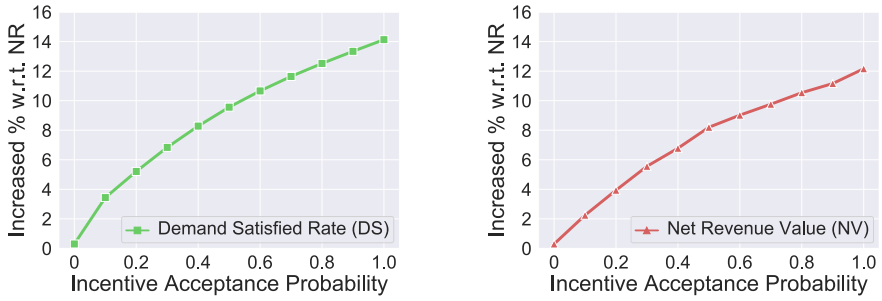
Figure 5.8: Performance of the proposed ac-PPO algorithm vs. (a) charging time, and (b) EV fully charged range.

approach (PPO-RND) produces worse results than the baseline NR. This is expected, as even within the optimal grid, the variations of station can be significant, where selecting a wrong station would hugely affect the reward. This would also have knock-on effects on training the PPO on top, since the obtained rewards can no longer faithfully represent the potential gain of the inter-grid actions. The revenue greedy approach (PPO-REV) is more sensible than PPO-RND, by offering 8% improvement in demand satisfied rate than the baseline NR. However, as discussed above, this approach tends to perform lots of unnecessary repositions and push EVs to high value sites, causing undesirable performance in net revenue. We observe similar trend in the demand gap greedy algorithm (PPO-DMD), which offers similar demand satisfied rate (about 8% improvement) and slightly better net revenue (2% improvement). As expected, the proposed ac-PPO performs the best overall, and the gap between ac-PPO and PPO-DMD is about 10% in net revenue and 6% in demand satisfied rate. This confirms that the two sub-actions (inter-grid and intra-grid) should be optimized jointly, and the proposed intra-grid policy network π^s_ϕ outperforms the rule-based baselines under the action cascading framework.

Impact of System Expansion Dynamics. This set of experiments investigate the impact of system expansion dynamics to the rebalancing algorithms. Here we only consider the state-of-the-art STRL and the proposed ac-PPO, as we have shown that the baselines are inferior to both of them in previous experiments. In the experiments, we adjust the simulator to allow different speeds of station network expansion, i.e., on average how many new stations should be deployed and existing stations closed per day. Essentially here we control the level of dynamics in the station network. We vary the speed from 0 to 3, where 0 means the station network is static, and 1 means station network expands at the same speed with that in the

real world. As shown in Fig. 5.7, we see that when there is no dynamics at all, the gap between STRL and ac-PPO is only about 4% in demand satisfied rate, and 6% in net revenue. Also we find that in this case, on average STRL only needs to reposition 1.7 EVs to satisfy an extra order, which is already quite efficient. This is expected, as in this static case STRL clusters the stations into groups, and uses spatial-temporal RL to estimate the future demand of those groups, in order to reposition the EVs accordingly. However, as the system begins to expand, the performance of STRL drops immediately. We already see that at the normal speed, the gap between STRL and our ac-PPO is more than 6% in demand satisfied rate, and 11% in net revenue. In the extreme case where the expansion speed is 3x, we see that gap in demand satisfied rate becomes almost 10%, while STRL can't increase the net revenue when the expansion speed is faster than 1.5. This is also expected, as STRL relies heavily on the station clustering performance, where as the station network is very dynamic, naturally its clustering algorithm would fail to produce optimal results, leading to inferior decisions in rebalancing. On the other hand, we see that the ac-PPO approach is very robust as the expansion speed increases, confirming that the proposed action cascading can work well under different levels of expansion dynamics.

Performance vs. Charging Time. In this set of experiments, we study a practical problem in the EV sharing industry: how charging time would affect our rebalancing performance. This is of great importance since one of the key problems of the current EVs is that the the range and charging delays often impact their usage patterns. For instance, the users may behave very differently when driving EVs whose batteries can be replaced immediately, those with super charging, or the ones with normal charging time. In this experiment, we first fix the EV range at 150km when fully charged, and vary the charging time of the EVs from 0 to 600min, where 0 in this case means the batteries of the EVs can be changed instantly. Note that in all other experiments, we assume the charging time of the EVs is 300min, which is consistent with the real data. Fig. 5.8a show the demand satisfied rate (DS) and net revenue (NV) increased by the proposed ac-PPO algorithm with respect to baseline NR at different charging rate. We see that clearly as charging time increases, the performance gain of our algorithm drops. This is expected because we can't perform any reposition action when the EV is charging. In the extreme case if the EVs are battery replaceable, the increase in NV is about 3% comparing to the standard case with 300min charging time. This means the approach of replaceable batteries does have its merits in some cases, and should be considered in practice. On the other hand, even in the slowest charging case (600min), our ac-PPO can still improve



(a) (b)

Figure 5.9: Performance of the proposed ac-PPO algorithm vs incentive acceptance probability: (a) increased % in DS, and (b) increased % in NV.

¿10% in NV and about 14% in DS. The gap between the cases of fastest and slowest charging is negligible in DS, and about 3% in NV. This means our ac-PPO algorithm is very robust to different charging time: for rebalancing task, even systems with slow chargers could enjoy considerable performance boost.

Performance vs. Battery Capacity. This set of experiments studies the impact of battery capacity i.e., fully charged range of the EVs to the performance of the rebalancing task. This is also a practical problem, which essentially indicates how shared e-mobility systems with different EV models (short range vs. long range) would behave under the proposed algorithm. Here we fix the charging time at 300min and vary the EV range from 75km to 225km. Note that in all the other experiments we use EV range as 150km. Fig. 5.8b shows the increased demand satisfied rate (DS) and net revenue (NV) of the proposed ac-PPO algorithm compared to the baseline NV. We can see that as the range of the EVs increases, the performance gain becomes more significant. This makes sense because EVs with longer range require less frequently charging, and often allow more flexible rebalancing: they could be repositioned to further stations if needed. We also observe that the performance is more sensitive for EVs with shorter range. For instance, when the range drops from 150km to 75km, the performance gain in NV is halved. However, even in that case our ac-PPO algorithm offers about 10% improvement in demand satisfied rate, as well as >5% more net revenue value, which is still better than the state-of-the-art. On the other hand, we see that after the range increases over 175km, the extra benefit brought by longer range becomes negligible. This means in practice, EV models with different ranges do react differently to the rebalancing task, but after a certain point the longer range EVs won't contribute much to the performance gain.

Performance vs. Incentive Acceptance Probability. The last set of experiments studies the rebalancing performance of the proposed ac-PPO algorithm under different incentive acceptance probability p. In our simulator, this probability p indicates how likely the users would agree to reposition the EVs to alternative stations, i.e., how cooperative they are. We vary p from 0 to 1, where 0 means the users decline any rebalancing offer, which is equivalent to the No Rebalancing (NR) as discussed in Section 5.5.2, and 1 means the users would accept all the reposition proposals. Fig. 5.9 shows the performance of our ac-PPO algorithm under different p values. We see that as the incentive acceptance probability increases, both DS and NV improves. This is expected since as the users become more cooperative, our algorithm can reposition more EVs, and thus smooth the vehicle distribution especially around the busy stations, enabling the system to satisfy more orders in the future. We also observe that as p increases, the gain in DS and NV becomes smaller, e.g., as p is larger than 0.5, the increase of performance is slowed down. This also makes sense because when the users tend to reject rebalancing (e.g., when p vary from 0 to 0.2), the performance of the system is limited by the amount of effective rebalancing operations carried out (i.e., those accepted by the users), while if the users are cooperative enough, it becomes less significant. On the other hand, this also shows that our algorithm is robust to different user models, e.g., as we can see in Fig. 5.9, even only 50% of the balancing operations are carried out (accepted by the users), we can still achieve 10% improvement in demand satisfied rate, and about 8% in net revenue.

5.6 Conclusion

In this chapter, we study the incentive-based rebalancing for fast expanding shared e-mobility systems. We formulate the rebalancing task as a Multi-agent Reinforcement Learning (MARL) problem, and solve it with the proposed approach of policy optimisation with action cascading. We design a simulator to simulate the real operation of the shared e-mobility systems, and calibrate the simulator with real data from an actual shared e-mobility system for over a year. Extensive experiments have shown that: 1) the proposed approach significantly outperforms the baselines and the state-of-the-art STRL in both metric of satisfied demand rate and net revenue; 2) PPO produces the best inter-grid policy comparing to other RL methods; 3) For intra-grid policy the proposed policy network works much better than the rule-based alternatives; 4) The proposed approach is robust to different levels of system expansion dynamics, while STRL fails as the dynamics increase; 5) The proposed

approach performs consistently with different charging time and EV range, while shorter charging time and longer EV range typically lead to better performance, but only to a certain extend. For future work, we would like to explore the more realistic case where the system can be rebalanced by both incentivized users and the dedicated staff, while there are multiple different EV models in operation. However, in practice, it may be difficult to decide the right amount of the incentives. Here we did not take this problem into consideration, which can be further solved through intensive user studies.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The development of new technologies such as shared mobility has generated tremendous opportunities to transform the way people get around cities, and reshape their life styles in urban environments. In this thesis, we focus on building robust deep spatio-temporal learning techniques for urban shared mobility systems, in which a diversity of entities such as users, vehicles and infrastructure, interact with the physical world and generate heterogeneous data footprints across spatial and temporal scales. More specifically, the thesis developed new modelling, optimisation and actuation approaches for the shared mobility systems, which aims to better understand the process, structure and evolution of mobility in our cities. In summary, this thesis contributed the following three main strands of research: demand forecasting, infrastructure optimisation and fleet management.

Demand Forecasting. This body of work has been presented in Chapter 3, which aims to develop new techniques to better model and predict the patterns of urban mobility across space and time. A key assumption that most of the existing work relies on, is that the mobility infrastructure has been fully deployed and thus static, while the emerging mobility systems in the real world, e.g., charging facilities for electric vehicles, may still continuously expand at visible speeds, posing new challenges for tasks such as demand or flow prediction. To capture such dynamicity, we propose new representation methods to model mobility patterns, where the spatial correlations within the evolving mobility systems are described using time-varying graphs, while local temporal dependencies are encoded with sequential models such as recurrent neural networks. On top of that, our work proposes a dynamic Graphical Convolutional Neural Networks (GCNs) architecture which is

more flexible, i.e., can handle data defined in non-Euclidean domains, and thus able to offer much better performance than the existing solutions by fusing information from heterogeneous sources and at various levels. The developed techniques have been adopted by a major shared e-mobility provider in one of the biggest markets to support demand forecasting and decision making in their operation.

Infrastructure Optimisation. This research thread has been discussed in Chapter 4, which investigates how infrastructure for shared mobility systems can be optimised in their deployment and operation across the spatio-temporal domain, to provide better service while being cost-effective. Instead of optimising such systems *in vivo*, we develop high fidelity simulations for mobility systems, which are calibrated with multi-modal data from the real world. The simulation environment serves as the "digital twins" of the actual systems, in which the impact of different deployment and operation strategies can be evaluated, enabling search algorithms to discover better strategies via trail-and-error. This is still challenging, as for systems operating at the city scales, the search space for optimal strategies can be complex and prohibitively large. To address that, we design a deep neural search algorithm, which employs a novel hierarchical controller to iteratively propose possible infrastructure deployment plans at different granularity, which are evaluated in the simulation environment. The results are propagated back as rewards to the controller, whose parameters are updated accordingly, steering it to generate better strategies in the future iterations. It has been shown that the proposed neural search algorithm can discover superior strategies for mobility infrastructure deployment than those provided by human efforts and the state-of-the-art, offering significant improvement in performance.

Fleet Management. This work has been detailed in Chapter 5, which studies the problem of fleet management in shared mobility systems, one of the fundamental tasks to improve the usability and performance of such platforms. The key challenge is that how to re-position the inevitably skewed distribution of fleet across the infrastructure to balance immediate and future returns, without sacrificing user experience. This becomes even more challenging when the mobility infrastructure itself is evolving, i.e., the fleet re-balancing strategies should take the infrastructure dynamics into account and adapt accordingly. In light of this, our work considers user-incentive based fleet management, which offers monetary rewards to users of the mobility systems in exchange for their cooperation to re-position vehicles at locations that are not their original destinations, but are more beneficial for future system operations. In particular, we formulate the fleet management problem in urban mobility as a multi-agent reinforcement learning task, where the strategies

of rebalancing actions, i.e., when and how to offer incentives, are learned from the interactions between agents and the simulated mobility systems. To cope with the dynamics in mobility infrastructure, we design a new action cascading technique, enabling the RL algorithms to work under varying action space, where the learned policies demonstrate superior performance and efficiency.

6.2 Future work

There are several future directions can be sketched based on the work of this thesis. Although we have contributed new deep spatio-temporal learning approaches for the modelling, optimisation and actuation of dynamic urban shared mobility systems, there are still underlying limitations and areas for future investigation. Concretely, we have identified the following directions that could potentially enhance and extend our work in the future:

Station-less Shared Mobility Systems. In this thesis, we primarily focus on station-based shared mobility systems. As discussed in Chapter 2, recently the station-less shared mobility systems are becoming popular in urban environments, from dockless shard bikes, to shared e-scooters and vehicles. Unlike the station-based systems considered in this thesis, which needs stations to dock/park vehicles/bikes, in those systems users can pick up or return the vehicles wherever available, offering more flexibility than the station-based ones. Therefore, it would be interesting to investigate if the approaches proposed in this thesis can be extended or adapted for station-less shared mobility systems, especially for the demand forecasting problem. If we can accurately predict the future demand at any particular location for the dockless shared mobility systems, it can help the system operators more efficiently manage the system such as designing dynamic pricing strategy to avoid parking congestion. It can also provide some insights for the city planners such as designing the bike-line. The most challenging part compared to station-based shared mobility system is that there is no fixed location (stations) in the dockless shared mobility systems. The potential solution can be taking the road segments as stations to predict the demand on each road segment or dividing the city into grids and treat the grids as stations. By doing this, the proposed demand forecasting approach in this thesis can be adapted for the dockless shared mobility systems.

Knowledge Transfer Between Systems. Specifically, this thesis considers learning to operate and actuate a single shared mobility system. In that sense, we collect data from a specific system, and use such data to obtain knowledge that can improve the operation of the very same system. In practice, it would be even more

appealing, if the knowledge learned from one system can be transferred to, and thus benefit the other systems that might not even in the same city. However, it is not always easy to collect a large set of data due to the privacy and commercial concerns, as well as the long time needed to collect the data, especially for a new system deployed in a new city. Therefore, an interesting future direction to explore is how the learned models, e.g., the GCN proposed in this thesis, trained with data from one system, can be carefully fine-tuned and transferred to other systems, e.g., forecasting the mobility demand of systems in a different city. However, it is not a trivial task as cities are different from each other due to the different public transportation services, different lifestyle of citizens and more. Therefore, it is very hard to build an end-to-end model to directly generate results in the new city. There are lots of opportunities can be explored along this direction. The knowledge transfer between systems can give suggestions to government or enterprises in the new city to make decisions in advance such as choosing station locations and planning public chargers.

Comprehensive User Study. Finally, another line of work that could potentially complement the contributions in this thesis is a comprehensive user study for the urban shared mobility systems. In practice, a comprehensive qualitative and quantitative user study towards the shared mobility systems in urban settings could provide key insights into the usage patterns of those systems, e.g., when, where and how users would like to use them, which in turn may help us to design better systems in the future. For instance, the incentive-based rebalancing approach discussed in this thesis could benefit from such a study in various ways. It is non-trivial to decide the right incentives should be provided to users because it may happen cheating events if the incentives are much more than their expectation, while users may not accept the incentives if it is far less than their expected rewards. Another difficulty is that the expected incentives of users are different. Even the same user under different travel purposes, the expected reward to help rebalancing the vehicles are different. Therefore, it can be very useful to survey a large amount of users from different age groups, occupations and travel intentions such that the system operators can design more effective incentive strategies and discovering more sensible reposition destinations, and thus offer better user experiences in general.

Bibliography

[1] F.-J. Van Audenhove, O. Korniichuk, L. Dauby, and J. Pourbaix, "The future of urban mobility 2.0: imperatives to shape extended mobility ecosystems of tomorrow," 2014.

[2] M. Nunu, R. Nausedaite, K. Eljas-Taal, K. Svatikova, and L. Porssch, "Study to monitor the economic development of the collaborative economy at sector level in the 28 eu member states," *European Commission: Brussels*, 2018.

[3] "Bluecity car sharing," https://www.blue-city.co.uk/, 2019, accessed: 2019-04-29.

[4] "Volkswagen starts 'we share' e-mobility car sharing in berlin," https://www.volkswagenag.com, 2019, accessed: 2019-04-29.

[5] "Bluesg," https://www.bluesg.com.sg/, 2019, accessed: 2019-04-29.

[6] "National travel survey factsheets," https://www.gov.uk/government/publications/nts-factsheets, 2018, accessed: 2021-02-06.

[7] "London datastore," https://data.london.gov.uk, 2018, accessed: 2021-02-06.

[8] "Nyc open data," https://opendata.cityofnewyork.us, 2012, accessed: 2021-02-06.

[9] "The gaia initiative," https://outreach.didichuxing.com/research/opendata/en/, 2018, accessed: 2021-02-06.

[10] B. Du, Y. Tong, Z. Zhou, Q. Tao, and W. Zhou, "Demand-aware charger planning for electric vehicle sharing," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1330–1338.

[11] D. Chai, L. Wang, and Q. Yang, "Bike flow prediction with multi-graph convolutional networks," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2018, pp. 397–400.

[12] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1774–1783.

[13] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *The World Wide Web Conference*, 2019, pp. 983–994.

[14] "ev-simulator," https://github.com/manluow/ev-simulator, 2021.

[15] M. Foth, *Handbook of research on urban informatics: the practice and promise of the real-time city: the practice and promise of the real-time city*. IGI Global, 2008.

[16] A.-N. Qazi, Y. Nara, K. Okubo, and H. Kubota, "Demand variations and evacuation route flexibility in short-notice bus-based evacuation planning," *IATSS research*, vol. 41, no. 4, pp. 147–152, 2017.

[17] C. Chen, D. Zhang, N. Li, and Z.-H. Zhou, "B-planner: Planning bidirectional night bus routes using large-scale taxi gps traces," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1451–1465, 2014.

[18] X. Kong, M. Li, T. Tang, K. Tian, L. Moreira-Matias, and F. Xia, "Shared subway shuttle bus route planning based on transport data analytics," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1507–1520, 2018.

[19] K. T. Seow, N. H. Dang, and D.-H. Lee, "A collaborative multiagent taxi-dispatch system," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 607–616, 2009.

[20] F. Miao, S. Han, S. Lin, Q. Wang, J. A. Stankovic, A. Hendawi, D. Zhang, T. He, and G. J. Pappas, "Data-driven robust taxi dispatch under demand uncertainties," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 1, pp. 175–191, 2017.

[21] B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang, "Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset," in *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE, 2011, pp. 63–68.

[22] D. Zhang, L. Sun, B. Li, C. Chen, G. Pan, S. Li, and Z. Wu, "Understanding taxi service strategies from taxi gps traces," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 123–135, 2014.

[23] M. Qu, H. Zhu, J. Liu, G. Liu, and H. Xiong, "A cost-effective recommender system for taxi drivers," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 45–54.

[24] "Vancouver guide to ride hailing in b.c." https://vancouversun.com/news/local-news/vancouver-guide-to-ride-hailing-in-b-c, 2020, accessed: 2021-03-03.

[25] "Ride hailing services," https://en.wikivoyage.org/wiki/Ride_hailing_services, 2020, accessed: 2021-03-03.

[26] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, and J. Ye, "Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 905–913.

[27] B. Zhao, P. Xu, Y. Shi, Y. Tong, Z. Zhou, and Y. Zeng, "Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 2245–2252.

[28] M. Asghari and C. Shahabi, "Adapt-pricing: a dynamic and predictive technique for pricing to maximize revenue in ridesharing platforms," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2018, pp. 189–198.

[29] M. Asghari, D. Deng, C. Shahabi, U. Demiryurek, and Y. Li, "Price-aware real-time ride-sharing at scale: an auction-based approach," in *Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2016, pp. 1–10.

[30] "Capital bikeshare," https://www.capitalbikeshare.com, 2021, accessed: 2021-02-29.

[31] "velib metropole," https://www.velib-metropole.fr/en_GB, 2021, accessed: 2021-02-29.

[32] "Zipcar flex," https://www.zipcar.co.uk/electric, 2019, accessed: 2019-04-29.

[33] L. Chen, D. Zhang, G. Pan, X. Ma, D. Yang, K. Kushlev, W. Zhang, and S. Li, "Bike sharing station placement leveraging heterogeneous urban open data," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, pp. 571–575.

[34] J. Liu, Q. Li, M. Qu, W. Chen, J. Yang, H. Xiong, H. Zhong, and Y. Fu, "Station site optimization in bike sharing systems," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 883–888.

[35] M. Kaspi, T. Raviv, and M. Tzur, "Parking reservation policies in one-way vehicle sharing systems," *Transportation Research Part B: Methodological*, vol. 62, pp. 35–50, 2014.

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

[37] J. Liu, L. Sun, W. Chen, and H. Xiong, "Rebalancing bike sharing systems: A multi-source data smart optimization," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1005–1014.

[38] "Mobike," https://mobike.com/global/, 2021, accessed: 2021-02-29.

[39] Z. Liu, Y. Shen, and Y. Zhu, "Where will dockless shared bikes be stacked? —parking hotspots detection in a new city," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 566–575.

[40] L. Pan, Q. Cai, Z. Fang, P. Tang, and L. Huang, "A deep reinforcement learning framework for rebalancing dockless bike sharing systems," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1393–1400.

[41] J. Bao, T. He, S. Ruan, Y. Li, and Y. Zheng, "Planning bike lanes based on sharing-bikes' trajectories," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1377–1386.

[42] T. He, J. Bao, R. Li, S. Ruan, Y. Li, C. Tian, and Y. Zheng, "Detecting vehicle illegal parking events using sharing bikes' trajectories." in *KDD*, 2018, pp. 340–349.

[43] S. Carley, R. M. Krause, B. W. Lane, and J. D. Graham, "Intent to purchase a plug-in electric vehicle: A survey of early impressions in large us cites," *Transportation Research Part D: Transport and Environment*, vol. 18, pp. 39–45, 2013.

[44] G. Wang, X. Chen, F. Zhang, Y. Wang, and D. Zhang, "Experience: Understanding long-term evolving patterns of shared electric vehicle networks," in *MobiCOM*, 2019.

[45] A. Sarker, H. Shen, and J. A. Stankovic, "Morp: Data-driven multi-objective route planning and optimization for electric vehicles," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 4, pp. 162:1–162:35, Jan. 2018.

[46] L. Yan, H. Shen, Z. Li, A. Sarker, J. A. Stankovic, C. Qiu, J. Zhao, and C. Xu, "Employing opportunistic charging for electric taxicabs to reduce idle time," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 1, pp. 47:1–47:25, Mar. 2018. [Online]. Available: http://doi.acm.org/10.1145/3191779

[47] G. Wang, W. Li, J. Zhang, Y. Ge, Z. Fu, F. Zhang, Y. Wang, and D. Zhang, "sharedcharging: Data-driven shared charging for large-scale heterogeneous electric vehicles," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, May 2019.

[48] Y. Gao, Y. Guo *et al.*, "Optimal planning of charging station for phased electric vehicle," *Energy and Power Engineering*, vol. 5, no. 04, p. 1393, 2013.

[49] Y. Xiong, J. Gan, B. An, C. Miao, and A. L. C. Bazzan, "Optimal electric vehicle charging station placement," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 2662–2668.

[50] Q. Liu, Y. Zeng, L. Chen, and X. Zheng, "Social-aware optimal electric vehicle charger deployment on road network," in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 398–407.

[51] C. Wei, Y. Wang, X. Yan, and C. Shao, "Look-ahead insertion policy for a shared-taxi system based on reinforcement learning," *IEEE Access*, vol. 6, pp. 5716–5726, 2017.

[52] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *2018 AAAI Conference on Artificial Intelligence (AAAI'18)*, 2018.

[53] U. Gupta, K. Jitkajornwanich, R. Elmasri, and L. Fegaras, "Adapting k-means clustering to identify spatial patterns in storms," 12 2016, pp. 2646–2654.

[54] T. K. Anderson, "Kernel density estimation and k-means clustering to profile road accident hotspots," *Accident Analysis & Prevention*, vol. 41, no. 3, pp. 359–364, 2009.

[55] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[56] "Support vctor machines," https://scikit-learn.org/stable/modules/svm.html, 2021, accessed: 2021-03-03.

[57] M. Kanevski, N. Gilardi, E. Mayoraz, and M. Maignan, "Environmental spatial data classification with support vector machines," 01 1999.

[58] X. Feng, X. Ling, H. Zheng, Z. Chen, and Y. Xu, "Adaptive multi-kernel svm with spatial–temporal correlation for short-term traffic flow prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2001–2013, 2018.

[59] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[60] M. A. Oliver and R. Webster, "Kriging: a method of interpolation for geographical information systems," *International Journal of Geographical Information System*, vol. 4, no. 3, pp. 313–332, 1990.

[61] Y. Cheng, X. Li, Z. Li, S. Jiang, and X. Jiang, "Fine-grained air quality monitoring based on gaussian process regression," in *International Conference on Neural Information Processing*. Springer, 2014, pp. 126–134.

[62] D. Nychka, S. Bandyopadhyay, D. Hammerling, F. Lindgren, and S. Sain, "A multiresolution gaussian process model for the analysis of large spatial datasets," *Journal of Computational and Graphical Statistics*, vol. 24, no. 2, pp. 579–599, 2015.

[63] J. Bijak, J. Hilton, E. Silverman, and V. D. Cao, "Reforging the wedding ring: exploring a semi-artificial model of population for the united kingdom with gaussian process emulators," *Demographic Research*, vol. 29, pp. 729–766, 2013.

[64] "Simple exponential smoothing," https://towardsdatascience.com/simple-exponential-smoothing-749fc5631bed, 2021, accessed: 2021-03-03.

[65] Y. Su, W. Gao, D. Guan, and W. Su, "Dynamic assessment and forecast of urban water ecological footprint based on exponential smoothing analysis," *journal of cleaner production*, vol. 195, pp. 354–364, 2018.

[66] H.-F. Yang, T. S. Dillon, E. Chang, and Y.-P. P. Chen, "Optimized configuration of exponential smoothing and extreme learning machine for traffic flow forecasting," *IEEE Transactions on Industrial informatics*, vol. 15, no. 1, pp. 23–34, 2018.

[67] "Arima for time series forecasting with python," https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/, 2021, accessed: 2021-03-03.

[68] U. Kumar and V. Jain, "Arima forecasting of ambient air pollutants (o 3, no, no 2 and co)," *Stochastic Environmental Research and Risk Assessment*, vol. 24, no. 5, pp. 751–760, 2010.

[69] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl *et al.*, "Constrained k-means clustering with background knowledge," in *Icml*, vol. 1, 2001, pp. 577–584.

[70] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[71] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," *arXiv preprint arXiv:1406.2984*, 2014.

[72] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*. IEEE, 2011, pp. 196–201.

[73] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 2013, pp. 8614–8618.

[74] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. ARTICLE, pp. 2493–2537, 2011.

[75] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[76] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *Journal of chemical information and modeling*, vol. 55, no. 2, pp. 263–274, 2015.

[77] T. Ciodaro, D. Deva, J. De Seixas, and D. Damazio, "Online particle detection with neural networks based on topological calorimetry information," in *Journal of physics: conference series*, vol. 368, no. 1. IOP Publishing, 2012, p. 012030.

[78] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[79] R. Vaillant, C. Monrocq, and Y. Le Cun, "Original approach for the localisation of objects in images," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 141, no. 4, pp. 245–250, 1994.

[80] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[81] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Scene parsing with multiscale feature learning, purity trees, and optimal covers," *arXiv preprint arXiv:1202.2160*, 2012.

[82] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[83] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[84] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[85] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[86] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proceedings of the 20th International Conference on Neural Information Processing Systems*, ser. NIPS'07, 2007, p. 161–168.

[87] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *arXiv preprint arXiv:1310.4546*, 2013.

[88] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*. IEEE, 2015, pp. 153–158.

[89] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "Lc-rnn: A deep learning model for traffic speed prediction." in *IJCAI*, 2018, pp. 3470–3476.

[90] T. H. Vu, L. Dung, and J.-C. Wang, "Transportation mode detection on mobile devices using recurrent nets," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 392–396.

[91] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.

[92] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[93] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *arXiv preprint arXiv:1606.09375*, 2016.

[94] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[95] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[96] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations (ICLR'18)*, 2018.

[97] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *2019 AAAI Conference on Artificial Intelligence (AAAI'19)*, 2019.

[98] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957.

[99] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[100] R. Bellman and R. Kalaba, "Dynamic programming and statistical communication theory," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 43, no. 8, p. 749, 1957.

[101] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[102] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, UK, 1994, vol. 37.

[103] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[104] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[105] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[106] A. Plaat, W. Kosters, and M. Preuss, "Model-based deep reinforcement learning for high-dimensional problems, a survey," *arXiv preprint arXiv:2008.05598*, 2020.

[107] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[108] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[109] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[110] Y. Li, Y. Zheng, and Q. Yang, "Dynamic bike reposition: A spatio-temporal reinforcement learning approach," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1724–1733.

[111] S. Shaheen, A. Cohen, and M. Jaffee, "Innovative mobility: Carsharing outlook," 2018.

[112] Y. Li, Y. Zheng, H. Zhang, and L. Chen, "Traffic prediction in a bike-sharing system," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015, p. 33.

[113] P. Hulot, D. Aloise, and S. D. Jena, "Towards station-level demand prediction for effective rebalancing in bike-sharing systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 378–386.

[114] J. Liu, L. Sun, Q. Li, J. Ming, Y. Liu, and H. Xiong, "Functional zone based hierarchical demand prediction for bike system expansion," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 957–966.

[115] Y.-C. Yin, C.-S. Lee, and Y.-P. Wong, "Demand prediction of bicycle sharing systems." 2014.

[116] W. Wang, "Forecasting bike rental demand using new york citi bike data," 2016.

[117] J. Froehlich, J. Neumann, N. Oliver *et al.*, "Sensing and predicting the pulse of the city through shared bicycling." in *IJCAI*, vol. 9, 2009, pp. 1420–1426.

[118] E. O'Mahony and D. B. Shmoys, "Data analysis and optimization for (citi) bike sharing." in *AAAI*, 2015, pp. 687–694.

[119] Z. Yang, J. Hu, Y. Shu, P. Cheng, J. Chen, and T. Moscibroda, "Mobility modeling and prediction in bike-sharing systems," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2016, pp. 165–178.

[120] L. Chen, D. Zhang, L. Wang, D. Yang, X. Ma, S. Li, Z. Wu, G. Pan, T.-M.-T. Nguyen, and J. Jakubowicz, "Dynamic cluster-based over-demand prediction in bike sharing systems," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016, pp. 841–852.

[121] L. Chen, D. Zhang, G. Pan, X. Ma, D. Yang, K. Kushlev, W. Zhang, and S. Li, "Bike sharing station placement leveraging heterogeneous urban open data," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 571–575.

[122] J. Sun, J. Zhang, Q. Li, X. Yi, and Y. Zheng, "Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks," *arXiv preprint arXiv:1903.07789*, 2019.

[123] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks," *Artificial Intelligence*, vol. 259, pp. 147–166, 2018.

[124] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926*, 2017.

[125] Y. Zheng, H. Zhang, and Y. Yu, "Detecting collective anomalies from multiple spatio-temporal datasets across different domains," in *Proceedings of the 23rd*

SIGSPATIAL international conference on advances in geographic information systems. ACM, 2015, p. 2.

[126] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi, "Crowd sensing of traffic anomalies based on human mobility and social media," in *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 2013, pp. 344–353.

[127] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[128] "Amap api," https://lbs.amap.com/api, 2019, accessed: 2019-04-29.

[129] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126 – 139, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0198971516303970

[130] "Openstreetmap api," https://www.openstreetmap.org, 2019, accessed: 2019-04-29.

[131] "Juhe api," https://www.juhe.cn/docs/api/id/277, 2019, accessed: 2019-04-29.

[132] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[133] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[134] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *Journal of transportation engineering*, vol. 129, no. 6, pp. 664–672, 2003.

[135] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 265–283. [Online]. Available: http://dl.acm.org/citation.cfm?id=3026877.3026899

[136] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[137] C. Chen, J. Liu, Q. Li, Y. Wang, H. Xiong, and S. Wu, "Warehouse site selection for online retailers in inter-connected warehouse networks," in *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017, pp. 805–810.

[138] B. Guo, J. Li, V. W. Zheng, Z. Wang, and Z. Yu, "Citytransfer: Transferring inter-and intra-city knowledge for chain store site recommendation based on multi-source urban data," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–23, 2018.

[139] J. C. García-Palomares, J. Gutiérrez, and M. Latorre, "Optimizing the location of stations in bike-sharing programs: A gis approach," *Applied Geography*, vol. 35, no. 1-2, pp. 235–246, 2012.

[140] Y. Li, J. Luo, C.-Y. Chow, K.-L. Chan, Y. Ding, and F. Zhang, "Growing the charging station network for electric vehicles with trajectory data analytics," in *2015 IEEE 31st International Conference on Data Engineering*. IEEE, 2015, pp. 1376–1387.

[141] C. Liu, K. Deng, C. Li, J. Li, Y. Li, and J. Luo, "The optimal distribution of electric-vehicle chargers across a city," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 261–270.

[142] Y. Xiong, J. Gan, B. An, C. Miao, and A. L. Bazzan, "Optimal electric vehicle fast charging station placement based on game theoretical framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 8, pp. 2493–2504, 2017.

[143] T. D. Barrett, W. R. Clements, J. N. Foerster, and A. I. Lvovsky, "Exploratory combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1909.04063*, 2019.

[144] Q. Cappart, E. Goutierre, D. Bergman, and L.-M. Rousseau, "Improving optimization bounds using machine learning: Decision diagrams meet deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1443–1451.

[145] J. Huang, M. Patwary, and G. Diamos, "Coloring big graphs with alphagozero," *arXiv preprint arXiv:1902.10162*, 2019.

[146] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[147] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," in *ICML*, 2017.

[148] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[149] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[150] O. Tremblay and L.-A. Dessaint, "Experimental validation of a battery dynamic model for ev applications," *World electric vehicle journal*, vol. 3, no. 2, pp. 289–298, 2009.

[151] "Lianjia," https://sh.lianjia.com, 2020, accessed: 2020-05-12.

[152] "Shanghai qiangsheng taxi data," https://sodachallenges.com/datasets/taxi-gps/, 2020, accessed: 2020-04-29.

[153] S. Ghosh, P. Varakantham, Y. Adulyasak, and P. Jaillet, "Dynamic repositioning to reduce lost demand in bike sharing systems," *Journal of Artificial Intelligence Research*, vol. 58, pp. 387–430, 2017.

[154] S. Ghosh, J. YuKoh, and P. Jaillet, "Improving customer satisfaction in bike sharing systems through dynamic repositioning," 2016.

[155] F. Kooti, M. Grbovic, L. M. Aiello, N. Djuric, V. Radosavljevic, and K. Lerman, "Analyzing uber's ride-sharing economy," in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 574–582.

[156] S. Jiang, L. Chen, A. Mislove, and C. Wilson, "On ridesharing competition and accessibility: Evidence from uber, lyft, and taxi," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 863–872.

[157] A. Singla, M. Santoni, G. Bartók, P. Mukerji, M. Meenen, and A. Krause, "Incentivizing users for balancing bike sharing systems," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[158] T. Raviv, M. Tzur, and I. A. Forma, "Static repositioning in a bike-sharing system: models and solution approaches," *EURO Journal on Transportation and Logistics*, vol. 2, no. 3, pp. 187–229, 2013.

[159] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A taxi order dispatch model based on combinatorial optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 2151–2159.

[160] C. Etienne and O. Latifa, "Model-based count series clustering for bike sharing system usage mining: a case study with the vélib'system of paris," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 5, no. 3, p. 39, 2014.

[161] D. Chemla, F. Meunier, T. Pradeau, R. W. Calvo, and H. Yahiaoui, "Self-service bike sharing systems: simulation, repositioning, pricing," *EURO Journal on Transportation and Logistics*, 2013.

[162] B. Bakker, S. Whiteson, L. Kester, and F. C. Groen, "Traffic light control by multiagent reinforcement learning systems," in *Interactive Collaborative Infor. Sys.* Springer, 2010, pp. 475–510.

[163] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[164] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[165] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[166] D. R. Hunter and K. Lange, "A tutorial on mm algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004.