INTEGRATING REINFORCEMENT LEARNING-BASED VEHICLE DISPATCH ALGORITHM INTO AGENT-BASED AUTONOMOUS TAXI FLEET SYSTEM SIMULATION

by

Zequn Li

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Industrial Engineering

School of Industrial Engineering

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Hua Cai, Co-Chair
School of Industrial Engineering

Dr. Vaneet Aggarwal, Co-Chair
School of Industrial Engineering

Dr. Samuel Labi
School of Civil Engineering

Approved by:
Dr. Abhijit Deshmukh

ACKNOWLEDGMENTS

Completing a second master thesis is an interesting experience. With another opportunity to re-think my methodology of scientific research, I feel more polished in terms of problem-solving. Definitely, it would be impossible without the support from various people. First, I would like to thank my co-advisors, Dr. Hua Cai and Dr. Vaneet Aggarwal. The meetings with them over the two years were very informative and inspiring, and I personally benefited a lot from them. Second, I would like to thank my committee members, including Dr. Samuel Labi and my co-advisors, for willing to be on my committee and offer their expertise, which further improved the quality of this study. Third, I would like to thank Abubakr O. Al-Abbasi and Mustafa Lokhandwala for their guidance. The intersection of our time at Purdue was tight, but it is very important in getting me started at the beginning of my research. Fourth, I would like to acknowledge all other uSMART members, including Zhaoyu Kou, Ruoxi Wen, Hao Luo, Li Song, Kendrick Hardaway, Zhuoli Yin, Runhua Sun, Yue Li, and Utkuhan Genc. West Lafayette is not a metropolis, but the stay with you was fun. Lastly, I would like to thank Dr. Abdollah Shafieezadeh and Dr. Rabi Mishalani, my co-advisors at Ohio State University when I was pursuing my first master’s degree. Without their prior mentorship, I could not have achieved this far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AV	Autonomous Vehicle
ABM	Agent-Based Model
DQN	Deep Q-Network
MOVI	Model-free Optimization of Vehicle dIspatching
NYC	New York City
RL	Reinforcement Learning
RMSE	Rooted Mean Squared Error
TNC	Transportation Network Company

ABSTRACT

On-demand mobility has drastically changed the way transportation systems are operated and greatly improved people's access to transportation services. Meanwhile, autonomous driving technology has matured over time, and driverless vehicles have already been operated in real life. Replacing traditional taxi fleet with reliable autonomous taxi fleet would improve the service quality of transportation systems even further. However, with a large fleet of fully controllable objects, the operational optimization of such system becomes challenging as well. Existing studies fail to address both the realisticness of system simulation and the advantage of optimization-based algorithms at the same time. To precisely measure the benefits of operating an AV taxi fleet, this thesis integrates a reinforcement learning algorithm into to an agent-based simulation model of a ride hailing system. A real-world scale simulation of New York City (NYC) taxi fleet is conducted, and the system performance with the algorithm is compared with the common rule-based and heuristic dispatch algorithms in relevant literatures. It was observed that (1) DQN dispatched vehicles conservatively but achieved similar rider service level with proactive dispatch methods; and (2) DQN outperformed all other dispatch methods evaluated in this study with significantly higher dispatch efficiency.

1. INTRODUCTION

Autonomous vehicles (AV) are receiving increasing attention as self-driving technologies being rapidly developed. By October 2020, more than 30 states have been testing or already permitted the operation of personal or commercial AVs on public roads under different regulations (Insurance Institute for Highway Safety, 2020). Using a fleet of fully automated (Level 5) AVs to provide mobility services have several potential benefits. First, since no driver is involved, vehicle operation in the fleet system is more controllable. AVs are expected to reduce accidents and the overall cost of operating an autonomous taxi fleet can be cheaper than that of a traditional taxi fleet due to the reduced labor costs (Martinez et al., 2015). Additionally, with vehicle-to-control-center and vehicle-to-vehicle connectivity, autonomous taxis can be managed more cooperatively as a system than human-driving taxis.

Many researchers have built models to quantify the potential impacts of autonomous vehicle adoption. Pavone et al. (2012) constructed a transportation system with autonomous vehicles serving riders who travel among a set of stations, which is analogized as the door-to-door style mobility service. They proposed a system operation model by balancing the idle vehicle supply at the station level with real-time system feedback and observed that it reduced the number of average waiting riders by more than ten times while doubling the vehicle dispatch trips than the naïve fix-rate fluid operation model (Korte and Vygen, 2008). Similarly, Zhang and Pavone (2016) formulated the autonomous on-demand mobility system as a closed Jackson network model and further proposed a system operation algorithm considering possible rider loss and road congestion. They tested the algorithm using a simulation experiment in the Manhattan and concluded that 8,000 autonomous vehicles (70% of the current taxi fleet size) are needed to serve the existing Manhattan trip demand without introducing additional traffic congestion. Moreover, Spieser et al. (2016) generalized the system operation algorithm by Zhang and Pavone (2016) to accommodate predictions of future riders and evaluated the benefit of enhanced rebalancing algorithm versus different fleet sizes using simulations of Seattle and Calgary. They proposed a framework to determine optimal fleet size as a trade-off of multiple system performance metrics and observed that the benefit of rebalancing is significant in both cities even for small fleet sizes. Wallar et al. (2018) estimated the regional rider arrival rates as non-homogeneous Poisson process and proposed a fleet rebalancing algorithm using integer linear programming. They tested their algorithm using simulations in Manhattan and concluded that 3,000 autonomous vehicles (with ride sharing) could reduce rider waiting time by 37% and unserved riders by 95% at the cost of increased fleet travel distance compared with the current 13,500 traditional taxi fleet. Similar prediction-based optimization methods, including spatiotemporal forecasting (Dandl et al., 2019) and long-short-term-memory (LSTM) neural network prediction (Iglesias et al., 2018), also consolidated the superiority of autonomous on-demand mobility systems over traditional ones. While these methods leveraged classic optimization approaches for fleet management, the nature of these algorithms is centralized and generally does not scale well versus the problem size (e.g., number of vehicles, resolution of stations/grids). In addition, these algorithms depend on the assumptions about the system (e.g., homogeneous riders' preferences, simplified vehicle movement and routing), which are not realistic enough to represent modern autonomous on-demand mobility systems.

To account for the heterogeneous rider demands and preferences, agent-based models (ABM) have been increasingly used to quantify the impacts of autonomous fleets. Researchers have modeled autonomous taxi operation in many cities such as Austin (Fagnant (2016)), Berlin (Maciejewski and Bischoff (2018)), Bloomington (De Souza et al. (2020)), Lisbon (Martinez and Viegas (2017)), New York City (Lokhandwala and Cai, 2018; Zhang and Pavone, 2016), Melbourne (Javanshour et al. (2019)), Singapore (Shen et al., 2018; Spieser et al., 2014), Stuttgart (Heilig et al. (2017)), and Zurich (Boesch et al. (2016)) and concluded that autonomous taxis can help reduce taxi fleet size, alleviate traffic congestion, and improve fleet efficiency. Fagnant and Kockelman (2014) built a hypothetical transportation system and served this system using private vehicles and AV. They concluded that one AV can replace about ten private vehicles on average while introducing 11% extra vehicle travel distance. They also suggested that hidden benefits such as vehicles' life-cycle management might further increase the value of AV fleet. Maciejewski and Bischoff (2018) compared the congestion effect of private vehicle system and autonomous taxi system in Berlin. They found that if all private vehicles were changed to autonomous taxis and the road capacity were doubled to accommodate the extra AV rebalance trips, there would be no congestion even in downtown Berlin. De Souza et al. (2020) compared the agent-based autonomous taxi simulations with and without vehicle rebalancing in Bloomington and observed that vehicle rebalancing lowered average rider waiting time and increased fleet service rate and coverage at the cost of extra vehicle travel distance. Martinez and Viegas (2017) simulated two fleet systems with different AV capacities and compared the system-level performances with the current Lisbon transportation system. Their results show that total CO_2 emissions can be reduced by 40% and system congestion can be reduced by 30% by completely adopting AV fleet. Lokhandwala and Cai (2018) conducted a comparative case study between traditional taxi fleet and autonomous taxi fleet in New York City using an agent-based model considering riders' heterogeneity acceptance to sharing and trip delays. They concluded that autonomous taxi fleet (with ride sharing) outperformed traditional taxi fleet by more than 50% in terms of fleet sizing, vehicle travel distance, and vehicle occupation under different scenarios, but their autonomous taxi fleet operation algorithm prioritized high-demand regions (Manhattan) while under-served other regions comparing with traditional taxi fleet. Javanshour et al. (2019) conducted an agent-based simulation of AV fleet system in Melbourne considering uncertain rider demands and concluded that about 84% of current private vehicles can be replaced by AVs at the cost of extra vehicle mileage. Shen et al. (2018) proposed a novel AV-Bus hybrid service system and analyzed the effect of this hybrid system in Singapore under different fleet sizes, rider preferences, and synergy modes using agent-based models. They found that this hybrid system outperforms the current system in terms of service level, transportation efficiency, and financial sustainability only in selected situations and further attentions are needed on system operation, regulation, and design. Heilig et al. (2017) analyzed the benefits of autonomous taxis in the situation of multi-modal transportation systems in Stuttgart and observed that 85% of private vehicles can be replaced by autonomous taxis (with ride sharing) and that public transit usage also increased along with the adoption of autonomous taxis. Boesch et al. (2016) explored the autonomous taxi fleet sizing potentials using agent-based simulation of the current Zurich transportation patterns and concluded that 90% of current fleet can be replaced if the same trips were served by autonomous taxis with proactive vehicle rebalancing without significant increase of rider service level.

While these models have the merit of containing real-world details of road networks and rider demands, vehicle dispatching is often simplified based on predetermined rules. For example, Heilig et al. (2017) assumed that the next day's demand is perfectly known and calculated the minimal needed vehicles to serve the riders of the entire next day in each zone. Then, they added vehicles to zones lacking vehicles and did nothing to zones with extra vehicles only once during the midnight as vehicle dispatching and did not proactively relocate vehicles in-between. Fagnant and Kockelman (2014) and Fagnant (2016) meshed the service region at different levels, and vehicles heuristically searched for dispatch locations with the fewest net vehicle supply progressively from coarser to finer grids. De Souza et al. (2020) rebalanced the idle vehicles to region centroids so that the idle vehicle supply at each region centroid is proportional to a linear combination of the area of the region and the number of riders in the region. Martinez et al. (2015) and Martinez and Viegas (2017) navigated idle vehicles to the nearest taxi stand/bus station for a hybrid public and taxi transportation system. Huang et al. (2020) defined similar rules, but they relocated idle taxis to rail stations along a congestible road network. Although parking the idle taxis to nearby parking lots was intended to adjust idle vehicle supply, Yan et al. (2020) compared no relocation with relocation to the nearest parking lot as in Huang et al. (2020) and found that the average rider waiting time in the no relocation case was actually shorter. The authors expressed the view that this counterintuitive result was due to the additional congestion induced by the relocation trips. Lokhandwala and Cai (2018) stored five “preferred” waiting locations (sampled based on historical rider demand densities) for each vehicle and when vehicles become idle, they were dispatched towards one of their “preferred” locations (selected randomly from the list). These dispatching rules are summarized in Table 1.1.

Table 1.1. Summary of dispatch rules.

Reference	Dispatch Rule	Region	Results
Heilig et al. (2017)	Add vehicles during midnight and no dispatch during the day	Stuttgart	85% private vehicles can be replaced (with ride sharing); public transit usage increased.
Fagnant and Kockelman (2014)	Search to balance rider-vehicle mismatch among neighboring regions from coarser to finer level.	Artificial	One AV (with ride sharing) can replace ten private vehicles with 11% extra distance.
De Souza et al. (2020)	Move vehicles to pre-selected region centroids.	Bloomington	AV rebalancing reduces rider waiting time, increase service rate and coverage while increasing fleet travel distance.
Martinez and Viegas (2017)	Move vehicles to nearest taxi stand/bus station.	Lisbon	CO_2 emissions can be reduced by 40% and system congestion can be reduced by 30% when fully adopt AV.
Huang et al. (2020)	Move vehicles to nearest rail station.	Austin	Larger fleet size leads to more AV adoptions in first-mile last-mile transit connection.
Yan et al. (2020)	Move to nearest parking lot.	Minneapolis-Saint Paul	Vehicle relocation generates 8% more travel distance.
Lokhandwala and Cai (2018)	Randomly sample from rider demand.	NYC	AV drastically outperforms traditional taxi in fleet sizing, vehicle travel distance, and vehicle occupation.

Although these rules and heuristics usually are less computationally intensive and intuitively realistic approximations of real-life human dispatching strategies, they are generally less mathematically formulated and rigorous than optimization approaches and might not consistently and robustly function across different scenarios.

Recently, reinforcement learning (RL) algorithms have been introduced by the computer science community to optimize vehicle dispatching. Instead of modeling the transportation system with specific mathematical structure and solving an optimization problem using the mathematical model (model-based approach), RL algorithms do not rely on strong assumptions on the specific dynamics of transportation systems and generally learn to adopt a strategy by consecutively interacting with decision-makers and gaining benefits from the decisions (model-free approach). Some recent studies show remarkable improvements in system efficiency by using distributed Deep Q-Network (DQN), a type of RL algorithm. Oda and Joe-Wong (2018) proposed a framework called Model-free Optimization of Vehicle dIspatch (MOVI), where they built a ride hailing fleet simulator and learned an optimal dispatch policy by fitting a neural network to gain long-term system benefits. They assumed that vehicles globally shared information of the fleet system via a dispatch center but independently and identically elicited rebalancing destinations without coordinating with each other. These setups make DQN distributed in nature thus scalable computation-wise, which greatly enables real-time feedback on a large system through leveraging the power of parallel computing. They observed that MOVI improves rider service rate by 20% and 76% compared with a benchmark long-term optimization case and a no-dispatch case, respectively. Al-Abbasi et al. (2019) further proposed a DeepPool framework, which generalized MOVI to ride sharing scenario, and concluded that the DQN algorithm outperforms centralized algorithms in system performances in both with and without ride-sharing systems. Wen et al. (2018) implemented the DQN algorithm in three different demand scenarios and showed that DQN overall reduced fleet size by 14% while inducing minor extra vehicle travel distances but was 2.5 times faster than the local anticipatory method while preserving similar system performance. Similar studies include the optimization of people-goods hybrid transportation system (Manchella et al., 2020), the joint-optimization of vehicle matching, pricing and dispatching (Haliem et al., 2020) and the pursuit of enhanced mathematical representation of the transportation system using similar framework (Holler et al., 2019).

Although these studies showed that DQN algorithms improved vehicle dispatching performance, the simulation models used in these studies generally are over-simplified compared to those agent-based models built by the researchers in the transportation community. For example, Al-Abbasi et al. (2019) and Oda and Joe-Wong (2018) both assumed that all riders enter the system at the beginning of a simulation time step and impatiently leave the system at the end of this time step if they are not matched with a vehicle. In real world scenarios, some riders could be more patient and are willing to wait longer for this matching process. Agent-based models, such as De Souza et al. (2020) and Lokhandwala and Cai (2018), typically set a deterministic waiting time threshold or use a stochastic distribution that spans multiple simulation time steps to reflect this waiting process and heterogeneous rider patience. Additionally, Al-Abbasi et al. (2019) and Oda and Joe-Wong (2018) also approximated the trip time along the road network with a pair-wise grid-to-grid trip time matrix for computation efficiency. Wen et al. (2018) even assumed vehicles move towards the destinations at a fixed speed without considering an actual road network. Because of the path dependency nature of the transportation system, inaccurate vehicle arrival times may lead to accumulated inaccurate vehicle availability information over the long simulation horizon, causing biased results. The models developed by the transportation community typically have more detailed representation of the road network and vehicle travel path and speed. For example, the model developed by Lokhandwala and Cai (2018) moved vehicles on high-fidelity road networks and estimated travel speed in different regions based on historical data. As a result, the rooted mean squared error (RMSE) of predicted travel time in Lokhandwala and Cai (2018) is 0.867 minutes, much lower than the 4.739 and 3.75 minutes of RMSE in Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019), respectively. Levin et al. (2016) further considered congestible road networks and accounted for the impact of traffic volumes on trip time. In terms of simulation resolution, Al-Abbasi et al. (2019) and Oda and Joe-Wong (2018) discretized the simulation into multiple time steps and updated the behavior and status of all vehicles and riders at the beginning of each time step. However, this procedure might induce bias, and the interval of each time step needs to balance simulation efficiency and realisticness. When the interval is too small, the frequent dispatching is not always necessary. When the interval is too large, decision input of the dispatching algorithm may not be updated in time, which can bottleneck the influence of dispatching algorithm on the system. By contrast, Lokhandwala and Cai (2018) modeled both vehicles and riders as independent entities with their own timelines of events, and the events of rider and vehicles are triggered by their own encountered events with a resolution to 0.01 second. This is a more accurate and realistic setup to the real-life transportation systems.

The system performance of an autonomous taxi fleet can be significantly affected by the vehicle dispatching algorithm, and evaluating the benefit of dispatching algorithms accurately in a realistic simulation environment as a testbed, is also crucial. For most RL literatures, the algorithms are evaluated using simplified simulators without detailed modeling of the transportation systems. Whereas in the realistic agent-based transportation models, the dispatching rules are not optimized. Implementing the RL algorithm in a realistic ABM simulator can potentially provide a solution to the above discussed limitations to better model an autonomous fleet.

In this thesis, the simulator by Lokhandwala and Cai (2018) was integrated with a distributed DQN algorithm for vehicle dispatching, and a new reward function definition was proposed. Specifically, the thesis aimed to optimize vehicle dispatch efficiency by maximizing the net dispatch benefit of each vehicle. The developed algorithm was tested using a ride hailing system of NYC as a case study and the DQN algorithm was compared with rule-based dispatching methods commonly used in agent-based AV modeling literatures.

The rest of this paper is organized as follows. Section 2 mathematically defines the dispatch problem and the framework of DQN algorithm. Section 3 describes the simulation setup of the case study. Section 4 presents the results. Section 5 concludes the study and discusses limitations and future work.

2. PROBLEM DEFINITION AND DQN FRAMEWORK

2.1 Problem Setup and Objective

This study evaluates a ride hailing system with AVs serving the travel demands. The system is simulated using an agent-based model adopted from Lokhandwala and Cai (2018) with the modification that vehicle rebalancing decision is formulated as a distributed reinforcement learning (RL) problem similar to the one proposed by Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019). The idle AV dispatching aims to optimize the net dispatching benefit (defined in more detail in Section 2.2.3) of the entire fleet by maximizing each individual AV's net dispatching benefit. The agent-based model includes three types of agents as shown in Figure 2.1:

1) **Riders** who enter the system at their trip origins at specific time according to historical trip data and request rides from vehicles to be delivered at their trip destinations (ride pooling is not considered in this study);
2) **Vehicles** that pick up riders from their trip origins and drop them off at their trip destinations while in service and relocate themselves to certain regions (based on rebalancing actions received from the control center) in anticipation of future demands while idle (in this study, all vehicles are assumed to be autonomous);
3) A **control center** that has real-time geographic information and status of all vehicles and ride-requesting riders. The control center dispatches all vehicles, matching them with riders to serve requested demands, relocating the vehicles for fleet rebalancing, and routes the vehicles to their destination. The control center is fully aware of current system information and sends the information to the reinforcement learning (RL) agent to make fleet rebalancing decisions.

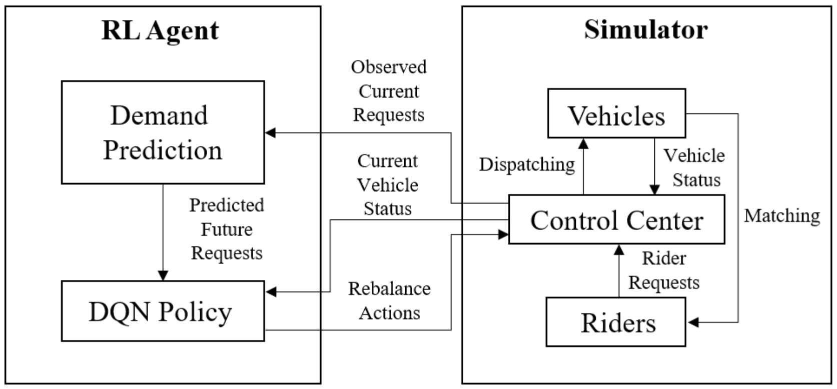

Figure 2.1. Model framework and the interactions between the agents-based simulation model and the reinforcement learning (RL) agent.

It may be noted that the control center serves as a platform to communicate with vehicles and to conduct fleet rebalancing. Here, the study separates the RL-related computation module as a RL agent from the control center to better represent the interactions between the algorithm and the simulation environment as in a general RL setup. Furthermore, it is assumed that all RL-related computations are conducted inside the RL agent, and the outcome of such computations are sent back to each vehicle via the control center. This setup does not appear to be distributed in architecture; however, the RL problem is actually solved in distributed manner, and the framework is equivalently distributed (each vehicle is making its own decision without central coordination). Also, for consistency and clarity, the study refers to the operation of moving idle vehicles to new locations as vehicle dispatching specifically from here onwards. Details of the interactions between the RL agent and the simulator are discussed in Section 2.3. Generally speaking, at each simulation step (e.g., 1 second), occupied vehicles will move along the identified route to continue delivering riders; idle vehicles that are close to rider requests will be matched with the requests and pick up the riders; and the other idle vehicles will be dispatched for rebalancing. Because too frequent vehicle redistribution does not benefit system efficiency, I define ΔT (e.g., 15 minutes) as the rebalancing interval similar to Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019) and discretize the simulation period into a time series $T = (T_0, T_0 + 1\times\Delta T, T_0 + 2\times\Delta T, \dots, T_0 + t\times\Delta T, \dots)$ with even spacing of ΔT starting from T_0 (ΔT is greater than the simulation step). At each dispatching interval (t), the control center receives system status information (s_t), sends the information to the RL agent to make dispatching decisions, and communicates the relocating actions (a_t) to idle vehicles that need to be rebalanced. After executing the relocating actions, the simulation of the fleet system progresses with vehicle and rider interactions until reaching the next dispatching interval ($t+1$) and obtain a new system status (s_{t+1}). At the beginning of this new interval, the dispatching decisions made in the previous interval are given rewards (r_t), which quantitatively measure the benefits gained by the dispatched vehicles during the period of ΔT (from t to $t+1$). The value of the reward is determined by a reward function $\boldsymbol{r}$(.) using s_t and a_t (more details in Section 2.2.3).

The complexity of jointly optimizing multiple vehicles scales exponentially with the number of vehicles, considering the possible permutations of vehicles' actions. Therefore, following the same setup as in Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019), I assume that the operation strategy of each vehicle is independent and identical, and the vehicles share the same knowledge on system transitions but not on each other's actions. The optimization of the entire fleet is therefore simplified to the optimization of multiple identical entities. Although no centrally managed coordination among the vehicles is considered, this distributed setup greatly boosts the computation efficiency which benefits overall system performance (Al-Abbasi et al., 2019; Oda and Joe-Wong, 2018).

Additionally, considering the path dependence nature of transportation systems (the feasible location of a vehicle in the next time step depends on the location of the vehicle at the current step), the long-term rewards of the system needs to be optimized, instead of just independently optimizing rewards at each time step. The long-term benefit can be defined as a Q value (Eq. (1), Sutton and Barto (2012)), which discounts future rewards exponentially and then measures the long-term benefit as the sum of all current and discounted future rewards.

$$Q_t = \sum_{j=t}^{\infty} \gamma^{j-t} r_j \tag{1}$$

Here $\gamma < 1$ is the discount factor of future rewards and r_j is the reward value at time step (dispatching interval) j. By maximizing Q_t rather than r_t at time step t, the effects of a_t on future system transitions are also included in the decision-making at time step t. However, due to the complexity of the system, the relationship between the Q value and the system's states (s_t) and actions (a_t) could not be explicitly described (Sutton and Barto (2012)).Without the explicit transition probabilities, it is hard to determine future states and actions and to further obtain future rewards. Therefore, instead of directly calculating the transition probabilities, deep neural networks (Q-network) are used to empirically estimate the expected Q value given current s_t and

a_t over possible future system transitions (Q-learning, Sutton and Barto (2012)). The Q-network can be denoted as $\widehat{Q_t}(s_t, a_t; \theta)$, where θ is the parameter set of the deep neural networks. Because calculating the future rewards over a variable and infinite time steps is not computationally viable (Sutton and Barto (2012)), a second Q-network is used to estimate the future reward (more details in Section 2.2.3). The entire algorithm is commonly referred to as Deep Q-Network (DQN) and the dispatching strategy learned by DQN is referred to as DQN policy.

The Q-networks need to be trained to learn the Q values given s_t and a_t by simulating the system operations and interactions iteratively until the RL agent learns an optimal and stable strategy. With the trained DQN, the dispatching decisions can be made based on choosing actions corresponding to the maximum Q value (DQN policy). Detailed definitions of s_t, a_t, r_t, and the framework of DQN are presented in Sections 2.2 and 2.3, and the agent-based simulation model is explained in Section 3.2.

2.2 Mathematical Definitions of State, Action, and Reward

2.2.1 State

Adopting the s_t definition and setup from Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019), the system state includes four vehicle-related variables and one rider-related variable: all idle vehicles, idle vehicles that are on the way to previously dispatched locations, vehicles that will become available within 1 ΔT, vehicles that will become available within 2 ΔT, and predicted future rider distribution. These five variables are formulated as matrices by gridding the system (to create variables of similar shapes as DQN inputs). Using the case study of New York City (NYC) as an example, the study area is defined as from 40.5° N to 40.95° N and from 74.1° W to 73.65° W (Figure 2.2). I define the study area to be slightly larger than the one in Oda and Joe-Wong (2018) (from 40.6003° N to 40.9003° N and 74.0407° W to 73.7501° W) to better cover the entire NYC and its outskirt. The main grid system comprises of 40×40 major grids (gray lines in Figure 2.2). The future rider demand is predicted using a finer system that comprises of 200×200 minor grids (too small to show) and then being aggregated into the 40×40 major grids. A random forest model is fitted using the observed rider distributions of the previous 1 ΔT and 2 ΔT as inputs to predict the future rider distribution within the next 2 ΔT. Each major grid is approximately 1.25km×1.25km, and each minor grid is approximately 250m×250m. Based on the grids, the count of different vehicles and riders in each grid during the specified time period is calculated to form the variable matrices. Combining all five variables together, s_t is described by a 40×40×5 tensor, which is the main input of DQN. In Figure 2.2, the vertical and horizontal gray lines are grid boundaries of the 40×40 major grids, and black lines are boundaries of NYC boroughs. For an example vehicle located in the red grid, its action space is the yellow grids (15×15 major grids centered at the vehicle's current location).

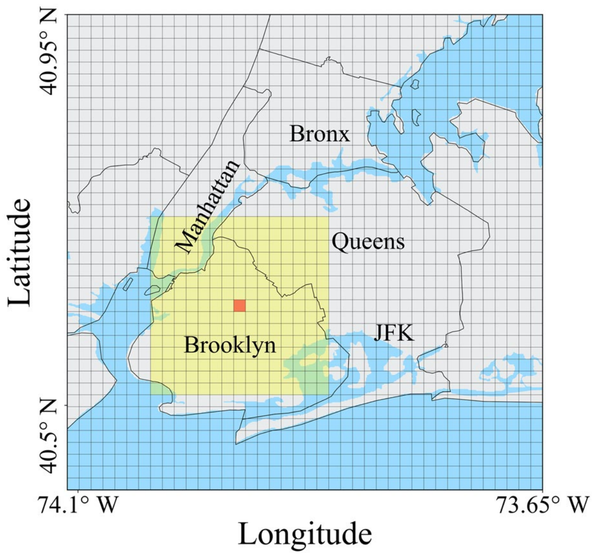

Figure 2.2. Major grid system.

2.2.2 Action

Similar to Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019), an idle vehicle can be dispatched within a sub-region (15×15 major grids, which is approximately 19km×19km) centered around its current location (as illustrated by the yellow grids for a vehicle located in the red grid in Figure 2.2). The size of the sub-region is determined based on the average distance that a vehicle can travel within 1 ΔT. I denote $a_{t,i}$ as the dispatch action of vehicle i at dispatching cycle t, and $A_{t,i}$ as the set of all possible dispatch actions of vehicle i at dispatching cycle t. Because some grids in the action space may not be reachable, such as geographically disjointed locations or water regions, I pre-determined all reachable grids through evaluating the feasibility of routing from locations in one grid to those in other grids based on OpenStreetMap. $A_{t,i}$ only includes reachable grids that are within the action space. Vehicles that have not yet reached the previously dispatched destinations are ineligible to be dispatched again.

A series of auxiliary variables are also needed to provide the context of the dispatch action so both local action space and their actual locations with respect to the entire region can be captured by Q-network. A total of 11 15×15 matrices are used to describe the day of week, hour of day, position, coordinate, move coordinate, distance, and sensibleness features. Four constant matrices are used to describe the sine and cosine value of the day of week and the hour of day (such transformation can better describe time than the original variables (Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019)). One zeros matrix with only the vehicle's center grid set to 1 documents the vehicle's current position in the action space. Two constant matrices are used to record the vehicle's current location in the global space, with another two matrices documenting the location of the vehicle after taking different actions. One matrix is used to document the distance of all grids in the action space to the center grid, which indicates dispatch cost; and one indicator matrix is used to record the sensibleness feature -- whether each grid in a_t is reachable or not. All elements in the auxiliary matrices are normalized to be within the range of -1 to 1. These 11 matrices are then concatenated into a 15×15×11 tensor as the auxiliary input of the Q-network.

2.2.3 Reward

Generally, one can expect a good dispatch action to timely relocate idle vehicles to locations near possible future requests so that the future riders are more likely to be served with shorter waiting time. In this study, to promote the benefit of being dispatched (reflected in the total duration providing service to riders) and demote the cost of dispatching (in terms of dispatch trip duration and time spent to pick up the next passenger), the study defined the reward received by an idle vehicle i at time step $t+1$ for the action taken in time step t as $r_{t,i}$, which is calculated using a reward function $\boldsymbol{r}$ as defined in Eq. (2).

$$r_{t,i} = \boldsymbol{r}(s_{t,i}, a_{t,i}) = \beta_0 + \beta_1 d_{t,i}^{(dispatch)} + \beta_2 d_{t,i}^{(pick-up)} + \beta_3 d_{t,i}^{(service)} \tag{2}$$

where $d_{t,i}^{(dispatch)}$ is the trip duration needed to reach the dispatched location given $a_{t,i}$; $d_{t,i}^{(pick-up)}$ is the time taken for the vehicle to travel from the dispatched location to the matched rider's pick-up location if one or more riders are matched to vehicle i between time step t and time step t+1 and zero if not matched; $d_{t,i}^{(service)}$ is the total duration of the trips that the vehicle served if one or more riders are matched to vehicle i between time steps t and t+1 and zero if not matched; and βs are the coefficients. I chose to use duration rather than distance in the reward function because trip time can better reflect the travel cost than distance in the case study city NYC (e.g., trips in Manhattan are often short in distance and but relatively long in duration due to congestion). The determination of βs is covered in Section 2.3.

2.3 DQN Framework

The Q-network needs to be trained before it can be used to generate effective dispatching actions. During the training phase, I start with a randomly initialized Q-network, run the simulation over the training period, and tune the Q-network parameters (θ, the weights and biases in every layer of the DQN architecture) using the observed system transitions described by the (s_t, a_t, r_t, s_{t+1}) tuple. I refer to such a system transition tuple as a piece of experience. At each dispatching cycle, one piece of experience is observed. To make more efficient use of the collected experience, the most recent pieces of experience are stored as experience memory ***D*** using a queue of fixed length. When optimizing the Q-network parameters, a batch of experience is randomly sampled from ***D*** (a.k.a. experience replay) to tune the parameters.

At the beginning of the training, the randomly initialized Q-network is a poor estimator of Q values, and only limited experience has been stored in ***D*** for the Q-network to learn. To explore possible new actions and exploit already observed actions, I use an ε-greedy strategy (Sutton and Barto, 2012) that dispatching vehicles to random locations with probability 1-ε and to "optimal" locations based on the current Q-network with probability ε. Based on the Q-network $\hat{Q}(s_t, a_t; \theta_t)$, the optimal action given the current system status s_t can be identified by computing the Q-values for all possible actions in the action space and choose the action that generates the greatest Q-value. The value of ε will be linearly increased from 0 to near 1 over time for the DQN policy to converge as more experience being observed.

While the reward gained from the actions at time t (r_t) can be directly obtained from the agent-based model, it is impossible to directly compute all future rewards. Therefore, a second Q-network ($\hat{Q}(s_{t+1}, a; \theta_t^-)$) is used as an approximator for future rewards. According to Eq. (1), Q_t can be viewed as the sum of r_t and γQ_{t+1}. Therefore, using $\hat{Q}(s_{t+1}, a; \theta_t^-)$ to estimate the Q_{t+1}value, the "true" value of Q_t can be estimated as the target Q-value (*TQ*) using Eq. (3).

$$TQ_t = r_t + \gamma \max_{a \in A_t} \hat{Q}(s_{t+1}, a; \theta_t^-) \tag{3}$$

For a given time step t, the analysis can compare the Q-value obtained by the current Q-network ($\hat{Q}_t$) and the estimated true value TQ_t and tune the Q-network's parameters (θ_t) to minimize the gap between the two. The gap is measured by the mean squared error between TQ_t and $\hat{Q}_t$, and is referred to as the loss function L_t (Eq. (4)). Given a non-empty experience memory, a batch of experience are sampled from the experience memory and set the value of θ_t to minimize L_t over the batch.

$$L_t(\theta_t) = E_{s_t,a_t,r_t,s_{t+1}}\left[\left(r_t + \gamma \max_{a\in A_t} \hat{Q}(s_{t+1}, a; \theta_t^-) - \hat{Q}(s_t, a_t; \theta_t)\right)^2\right] \tag{4}$$

Since the Q-network is constructed as a neural network, the minimization of L_t is achieved by performing backpropagation and gradient descent along the computation graph on θ. As $\hat{Q}(s_t, a_t; \theta_t)$ is trained to better estimate Q-value, the $\hat{Q}(s_{t+1}, a; \theta_t^-)$ also needs to be updated. Every certain time steps, the values of θ^- is replaced by the latest θ, while the values of θ^- is fixed in-between updates. Decoupling action generation and Q-value estimation through fixed target is proven to be effective in preventing overestimation of Q-values (Van Hasselt et al. 2016).

By progressively improving the quality of the estimator and choosing optimal actions accordingly, the DQN policy is learned to maximize the long-term benefits. The training is conducted using a consecutive series of simulations. Each simulation (1-day of system operation) is referred to as an episode. At the beginning of each episode, the simulator is reinitialized, but the Q-network parameters are preserved from the previous episode. At each time step, the agent-based model simulates the matching and movement of riders and vehicles and performs vehicle dispatching using DQN. After updating θ as described above at each dispatching cycle, DQN generates dispatch actions using the updated Q-network and sends them back to the simulator. Then, the simulator proceeds to the next time step and repeats the same process until the training phase ends. The training phase ends when the loss function defined in Eq. (4) and the average values of maximum Q-values generated by the Q-network for all vehicles at each time step stabilizes. The procedure of DQN training is summarized in Algorithm 1.

Algorithm 1: Deep Q-Learning with fixed targets and experience replay

```
1  Initialize an empty replay memory $D$, Q-network parameter $\theta$, and
     target Q-network $\theta^-$.
2  for $e = 0 : 1 : Episodes$ do
3      Initialize the fleet simulator with vehicles and rider requests.
4      for $t = T_0 : \Delta T : T$ do
5          Perform the dispatch and match order.
6          Update state $s_t$, reward $r_{t-1}$ based on action $a_{t-1}$.
7          Memorize system transition ($s_{t-1}$, $a_{t-1}$, $r_{t-1}$, $s_t$) in $D$.
8          Sample random system transitions ($s_i$, $a_i$, $r_i$, $s_{i+1}$) from $D$.
9          Set $a^*_{i+1} = argmax_a Q(s_{i+1}, a; \theta^-)$.
10         Set $TQ_i = r_i + \gamma Q(s_{i+1}, a^*_{i+1}; \theta^-)$.
11         minimize $(TQ_i - Q(s_i, a_i; \theta))$ w.r.t. $\theta$.
12         Set $\theta^- = \theta$ every $N$ steps.
13         Identify the set of idle vehicles $Idle_t$.
14         for vehicle $v$ in $Idle_t$ do
15             Choose $a_{t,i}$, with prob. $\epsilon$, a random action from $A_{t,i}$.
16             Else $a_{t,i} = argmax_a Q(s_t, a; \theta)$.
17             Send $a_{t,i}$ to vehicle $i$.
```

After the DQN has been trained, I run the agent-based model on other simulation days different from the training phase and fully follow the trained DQN policy without memorizing experience or modifying the Q-network. I compare the system performance of using the DQN policy to make dispatching decisions with those using rule-based dispatching to evaluate the benefits of integrating the RL agent into the model.

I use a similar Q-network architecture as Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019) as shown in Figure 2.3. Specifically, the main input s_t is of size 40×40×5. To condense the information that the main input passes to following layers, I apply three two-dimensional average pooling layers with pooling sizes of 7×7, 7×7 and 6×6 and strides all of 1×1, and this initial condensing procedure result in a tensor of size 23×23×5 and is denoted as main* input. Then, information is further condensed using a series of convolution layers. Three convolution layers of 16 5×5 filters, 32 3×3 filters, and 64 3×3 filters are sequentially followed. The auxiliary input of size 15×15×11, containing a_t, is then convolved using 16 1×1 filters and concatenated with the convolved main input before finally being fed into two additional convolution layers, one with 128 1×1 filters and the other one with 1 1×1 filter. The final output of DQN is a 15×15×1 matrix corresponding to the estimated Q values of every action $a_{t,i}$ in the action space $A_{t,i}$ given s_t. All convolution layers, except the last one, use tanh activation functions. At every iteration during training, I use RMSProp algorithm as the optimizer for DQN, and training data is passed to DQN with the batch size of 64. Table 2.1 summarizes the DQN parameters used in this paper and their values.

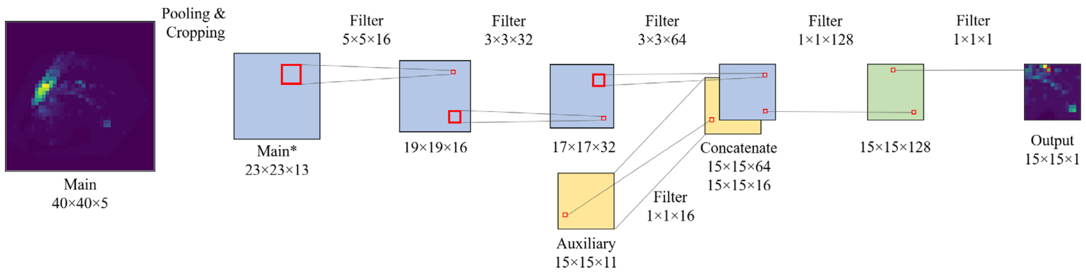

Figure 2.3. DQN neural network architecture and parameters of different layers.

Furthermore, I conduct a series of experiments determining the values of βs, which are levels of emphasize on different components, and set the search space of β_0 as 0, 1, 5, the search space of β_1, β_2 as 0, -1, -10, and the search space of β_3 as 0, 1, 10. I ran 42 cases of experiments in total and determined that setting the value of β_0, β_1, β_2 and β_3 to be 5, -1, -,1 and 1 is the optimal case considering the dispatch cost and benefit of the fleet system. For the rest of this paper, unless otherwise stated, these parameter values are used in the DQN algorithm. The results of the parameter selection are presented in the Appendix A Figure A.1.

A random forest model is fitted using the observed rider distributions of the previous 1 ΔT and 2 ΔT as input and the future rider distribution within the next 2 ΔT as output. Furthermore, this predicted future distribution is aggregated into a 40×40 matrix by summation with a stride of 5×5 grids. It may be noted that I used trip data of April 2014 as the training set and data of May 7, 2014 as the test set and achieved the RMSE of 1.571 and 1.663, respectively.

Because the training phase duration and time step interval in this study are different from Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019), I downscale the iteration related DQN parameters in this study proportionally to the ratio of training phase iteration in Oda and Joe-Wong (2018) and Al-Abbasi et al. (2019) and the training phase iteration in this study. Table 2.1 summarizes the DQN parameters used in this paper and their values.

Table 2.1. Summary of DQN parameters.

Parameter	Value
Discount factor γ	0.9
Learning rate of RMSProp	0.0025
Batch size of DQN training	64
Exploration steps of ε-greedy	300
ε at time step 0 in ε-greedy	0
ε at time step 300 in ε-greedy	0.95
Stored time steps of experience memory $\boldsymbol{D}$	300
Time steps of updating θ^- in fixed target	10

Since vehicles are initially assigned to random locations in the simulator at the beginning of the day and this random vehicle distribution is inconsistent with how vehicles typically operate during the day, I do not dispatch vehicles for the first hour of each day and allow vehicle movements to be fully driven by riders' travel patterns. By doing so, the starting system status at T_0 is more similar to real-world vehicle distributions when the vehicle dispatching in the remaining 23 hours is determined by DQN policy every ΔT minutes. I used 7 days for training and 1 day for evaluation. Therefore, the DQN training phase comprises 7×23×60/15 = 644 time steps (dispatching cycles), and the DQN evaluation phase comprises 1×23×60/15 = 92 time steps. The results of DQN training and evaluation are discussed in Section 4. The entire framework of DQN algorithm and simulator was executed on the high-performance computing community cluster operated by the research computing team at Purdue University.

3. SIMULATION SETUP AND CASE STUDY

3.1 Data

I set up the simulation model using New York City (NYC) as a case study and used NYC taxi trip records (NYC DOT, 2020) to represent the demands. This publicly available dataset provides detailed information of taxi trip data in NYC from 2009, including pick-up and drop-off date, time, location, trip distance, fare amount, and passenger counts, etc. Trip records of April 1-7 and May 7, 2014 were used for training and evaluation, respectively. I used 2014 data because they better represent taxi travel demands before the introduction of transportation network companies (TNC) such as Uber and Lyft.

3.2 Simulator

The agent rules and interactions in the system, modified from Lokhandwala and Cai (2018), is designed to be as follows.

1) Riders enter the system according to the raw trip data and wait to be matched with an idle vehicle. If a rider-vehicle pair is matched by the control center, the rider will wait to be picked up by the vehicle. If no match is found within the rider's maximum acceptable waiting time, the rider will leave the system unserved.
2) Vehicle activities, including matching and dispatching, are determined by the control center.
    a. Matching activities follow a deterministic algorithm. The closest idle vehicle within the rider's pick-up location is assigned to the rider. A maximum acceptable waiting time was also designed to reflect rider heterogeneity. This maximum waiting threshold for each rider was assumed to independently and identically follow a uniform distribution between 300 and 500 seconds similar to Lokhandwala and Cai (2018).
    b. Dispatch activities, generated by the DQN policy, occur at every ΔT. When a dispatch cycle is triggered, all eligible idle vehicles are directed to new destinations and start moving immediately or stay idle at their current locations. Moving vehicles, no matter occupied or not, are excluded from being dispatched or matched.

As an example, Figure 3.1 illustrates how riders and vehicles interact in this simulated system. When the vehicle is idle, it is given a new dispatch destination and moves toward the new location. Before reaching the assigned dispatched location, the vehicle is not allowed to be matched with a rider or to be dispatched to another location. Based on the vehicle's location and system state, it is possible that the vehicle is not assigned to another location while idle, but in this specific example, the vehicle is dispatched. Meanwhile, a rider enters the system and requests an idle vehicle through the control center. After the vehicle reaches the new location, it becomes idle again and waits to be matched by the control center, which matches the rider with the vehicle. The vehicle starts moving to the rider's pick-up location, and the matched rider waits to be picked-up by the vehicle. The ride-hailing service starts when the vehicle arrived at the pick-up location, and the vehicle, together with the rider, moves toward the rider's trip destination location to drop off the rider. When the vehicle reaches the drop-off location, the rider leaves the system, and the vehicle becomes idle again.

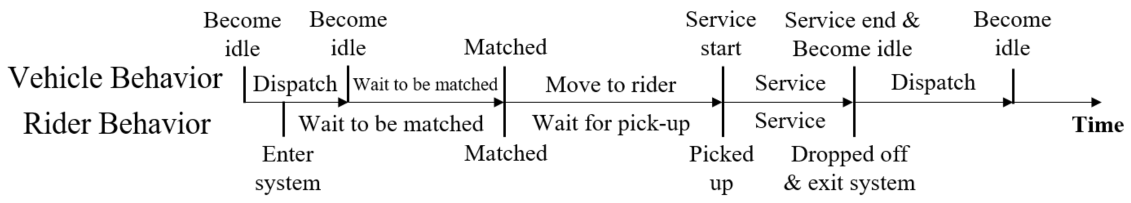

Figure 3.1. Example of how a vehicle is dispatched and then serves a rider.

Following the system dynamics defined above, there are three major components in vehicle and rider behaviors related to fleet system efficiency: vehicle dispatch, rider wait-for-pick-up, and service. In Section 4, I collected simulation logs and analyzed system efficiency from the perspectives of these three components, which are referred to as dispatch, pick-up, and service behaviors in the following discussions. The simulation model is programmed using AnyLogic software.

3.3 Benchmark Dispatch Rules

In this study, I used a fleet size of 8,000 AVs because this fleet size is sufficient to serve NYC taxi demands (Lokhandwala and Cai, 2018). I did not consider ride pooling in this study because the focus is on improving vehicle dispatching, but the developed model can be applied to study ride pooling in future research. To compare the dispatching performance of DQN policy with the heuristic rule-based dispatching, I developed five benchmark dispatching rules. All benchmark dispatching rules relocate idle vehicles every ΔT to have consistent dispatching cycle with the DQN policy.

To set the lower bound of the system performance, the "No-Dispatch" rule performs no rebalancing and assumes idle vehicles will park on roadside. Because the drop-off locations may not always have roadside parking available, the "Nearest-All" dispatch rule direct idle vehicles to the nearest parking lots. Common choices of parking lots are curbside parking slots (Yan et al. (2020)) and selected parking depots (De Souza et al., 2020; Huang et al., 2020; Martinez and Viegas, 2017). In this study, I considered all NYC curbside parking slots (NYC Open Data, 2016). For a centrally managed commercial AV taxi fleet, it is also possible to set up parking depots. The "Nearest-Cluster" dispatching rule directs idle vehicles to one of the 10 parking depots selected as the centroids of rider pick-up locations on the evaluation day using k-means algorithm (relocating idle vehicles to nearby parking depots where riders are more likely to appear). The distributions of "Nearest-All" and "Nearest-Cluster" parking lots are shown in the Appendix A Figure A.2 and A.3. The "Hotspot" dispatch refers to the heuristic dispatch algorithm modified from Lokhandwala and Cai (2018) that idle vehicles randomly choose from pre-sampled dispatch locations from the spatial frequency distribution of historical trip pickup locations.

I also implemented the heuristic dispatch algorithm by Fagnant and Kockelman (2014), denoted as "Hierarchical-Fill". There are four sequentially applied dispatch strategy (R1 to R4) in Hierarchical-Fill. The method starts with the most rider-vehicle imbalanced region (either with idle vehicles surplus or shortage) and then either moves surplus idle vehicles from surrounding regions to this region or move surplus idle vehicles in this region towards surround regions from coarser level (R1) to finer level (R2). After the initial rebalancing at macro level, Hierarchical-Fill spreads the densely distributed idle vehicles to surrounding local regions with no idle vehicles (R3 and R4) at micro level, aiming for faster rider pick up in these regions when there are requests. Note that I used the 40×40 major grid in this study as the finer grid as in the R2 strategy of "Hierarchical-Fill" and a 5×5 grid aggregated from the 40×40 major grid as the coarser grid in the R1 strategy of "Hierarchical-Fill". The coarser and finer grid are shown in the Appendix A Figure A.4.

Because these dispatch methods relocate idle vehicles differently, their impacts on the fleet system are different as well. To quantify and compare both the temporal and the spatial impact of dispatch trips on the fleet system, a set of system performance metrics are defined to measure the impacts of different dispatch methods. To compare vehicle performance of the fleet system, I summarized the distributions of dispatch, pick-up, and service trips, including the mean values and the total values. To compare rider service level of the fleet system, I summarized the rider service rate and rider mean waiting time. System-wise, I show the fleet traveled distance of different components and average in-operation proportion (average proportion of vehicle in-operation time out of the entire day) as measurement of fleet utilization. Moreover, to compare the temporal pattern of rider service level, I analyzed the time series of the number of served riders in NYC and the number of idle vehicles in Manhattan during the evaluation day, since most NYC trips are in Manhattan (Lokhandwala and Cai, 2018). Furthermore, to specifically quantify the efficiency of dispatch methods, I set No-Dispatch as the baseline and defined fleet dispatch efficiency of a dispatch method m as in Eq. (5).

$$Fleet\ Dispatch\ Efficiency_m = \frac{\#Served\ Riders_m - \#Served\ Riders_{No-Dispatch}}{Fleet\ Dispatch\ Distance_m} \qquad (5)$$

Here, the benefit of dispatch is measured as the number of extra served riders of method m compared with No-Dispatch and the cost of dispatch as the extra fleet dispatch distance using method m (the fleet dispatch distance of No-Dispatch is set to be zero). Generally speaking, the fleet dispatch efficiency is defined to be the average dispatch benefit over dispatch cost. Lastly, using the definition of fleet dispatch efficiency, I evaluated the trade-off of fleet dispatch efficiency and rider service rate of different dispatch methods.

4. RESULTS AND DISCUSSION

As mentioned in Section 2.3, I denote the average value of Q-network loss function as “loss” and the average value of maximum Q-values as “Q max”, and I determine whether the DQN policy stabilizes and converges by checking if Q max and loss change smoothly to their optimums. Figure 4.1 shows how the two learning curves progresses throughout the iterations in the training phase. The Q max curve increases rapidly to the value of 52 and remains at approximately 51 for the rest of the training. The loss curve gradually decreases from a very large number owing to the initialization to around 0.7 by the end of the training. The stability in learning curves suggests that the DQN parameters are reasonable.

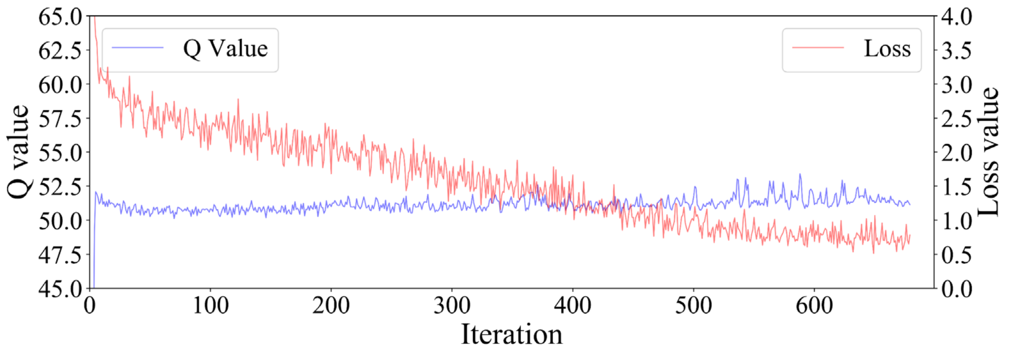

Figure 4.1. Q value and loss curves during training for optimal case.

I measure the system performance of the fleet system by inspecting the average behavior of vehicles and riders in the system during the evaluation day. Specifically, I define different metrics of describing vehicle service performance and rider service level. Overall, DQN dispatch introduced similar extra vehicle travel distance with the conservative algorithms (Nearest-All and Nearest-Cluster) but achieved similar rider service level with the proactive algorithms (Hierarchical-Fill and Hotspot). I explain the discrepancy of vehicle service performance and rider service level and compare the system-level performance of benchmarks and DQN in Sections 4.1, 4.2, and 4.3.

4.1 Vehicle Performance

As explained in Section 3.2, vehicle behavior includes three major components: dispatch, pick-up, and service. Table 4.1 summarizes the count, mean distance traveled by each vehicle, fleet total distance, and the proportion of total distance out of the fleet total travel distance for each component in the benchmarks and DQN. Note that continuous dispatch actions (i.e., being dispatched again after a previous dispatch without serving riders in between) are counted separately as two dispatching. Moreover, because the pattern of vehicle time and distance metrics are highly similar, I only present vehicle distance metrics.

Table 4.1. Vehicle performance comparison.

Dispatch Method	Fleet Traveled Distance (km)	Dispatch				Pick-up				Service		
		Count	Mean (km)	Total (km)	Proportion (%)	Count	Mean (km)	Total (km)	Proportion (%)	Mean (km)	Total (km)	Proportion (%)
No-Dispatch	1,771,996	0	0.00	0.00	0.00	343,410	0.99	339,976	19.18	4.17	1,432,020	80.82
Nearest-All	1,797,249	34,733	0.55	19,103	1.06	344,602	1.00	344,602	19.17	4.16	1,433,544	79.77
Nearest-Cluster	1,707,355	15,375	4.56	70,110	4.11	309,498	1.15	355,923	20.85	4.14	1,281,322	75.04
Hotspot	3,707,890	274,002	6.31	1,728,953	46.63	418,380	0.53	221,741	5.98	4.20	1,757,196	47.39
Hierarchical-Fill	2,613,284	158,060	4.79	757,107	28.97	411,569	0.56	230,479	8.82	3.95	1,625,698	62.21
DQN	1,972,327	19,202	3.95	75,848	3.85	400,101	0.83	332,084	16.84	3.91	1,564,395	79.31

Overall, the fleets following the Hotspot and Hierarchical-Fill dispatching rules travelled about twice of the distances than the other methods, and the dispatching trips are the major contributor of the extra travel in both cases with ten-times more total dispatch distance. The pattern for pick-up trips is opposite to dispatch trips. Hotspot and Hierarchical-Fill fleet spent less distance picking up riders than the other methods. However, the reduction of pick-up distance in Hotspot and Hierarchical-Fill is not large enough to compensate the extra distance caused by dispatching.

4.1.1 Vehicle Dispatch

For the dispatching component, the vehicle performance of Nearest-All is highly similar with No-Dispatch, because NYC curbside parking slots are widely distributed across the entire city and the additional dispatch trip is trivial (1.06% of total distance, 19,103km dispatch trip). Because the only difference among all methods is the dispatch component, both the pick-up and service component of Nearest-All are highly similar with those of No-Dispatch because of similar dispatch component. Nearest-Cluster and DQN promote fewer dispatch trips in number but longer trip in distance, and the fleet total dispatch distance of these two methods are about three times longer than Nearest-All. Despite the longer fleet dispatch distance of Nearest-Cluster and DQN compared with Nearest-All, they are still conservative dispatch methods with dispatch distance proportion less than 5%. By contrast, Hotspot and Hierarchical-Fill are much more proactive in vehicle dispatching. Hotspot fleet spent 1,728,953 km in dispatch trips, which is 46.63% of its fleet total travel distance, and Hierarchical-Fill fleet spent 757,107 km in dispatch trips, which is 28.97% of its fleet total travel distance. While the mean dispatch distance of Hotspot and Hierarchical-Fill are similar with conservative methods, the main reason of extra dispatch distance comes from the number of dispatch trips, which are larger by one order of magnitude. I illustrate the probability density distribution of dispatch trips' distance for each method in Figure 4.2. The dispatch trips of Nearest-All are observably shorter than all other methods, followed by DQN, Nearest-Cluster, Hierarchical-Fill, and Hotspot. The dispatch trip distance distributions of DQN and Nearest-Cluster are more concentrated than the rest and show slightly bimodal pattern, suggesting that there might be some typical dispatch modes.

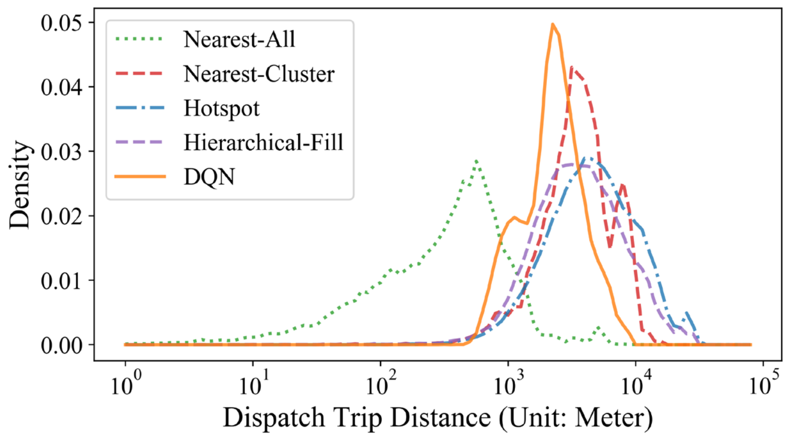

Figure 4.2. Probability density distribution of dispatch trip distance.

I further visualize the spatiotemporal distribution of dispatch origination and destination for each method. To be consistent with Lokhandwala and Cai (2018) and to better compare the spatiotemporal patterns, I divided a day into four periods: 0:00-7:00 (early morning), 7:00-15:00 (morning peak), 15:00-17:00 (afternoon), and 17:00-23:59 (evening peak). I chose to show the dispatch origination distribution during morning peak only as a representation of the entire day. Owing to the significant dispatch frequency differences among different dispatch methods, different color scales are used to represent the number of dispatches in each grid. Figure 4.3a to e show the dispatch origination distribution during morning peak for Nearest-All, Nearest-Cluster, Hotspot, Hierarchical-Fill, and DQN respectively. Similarly, Figure 4.3f to j show the dispatch destination distribution for corresponding methods.

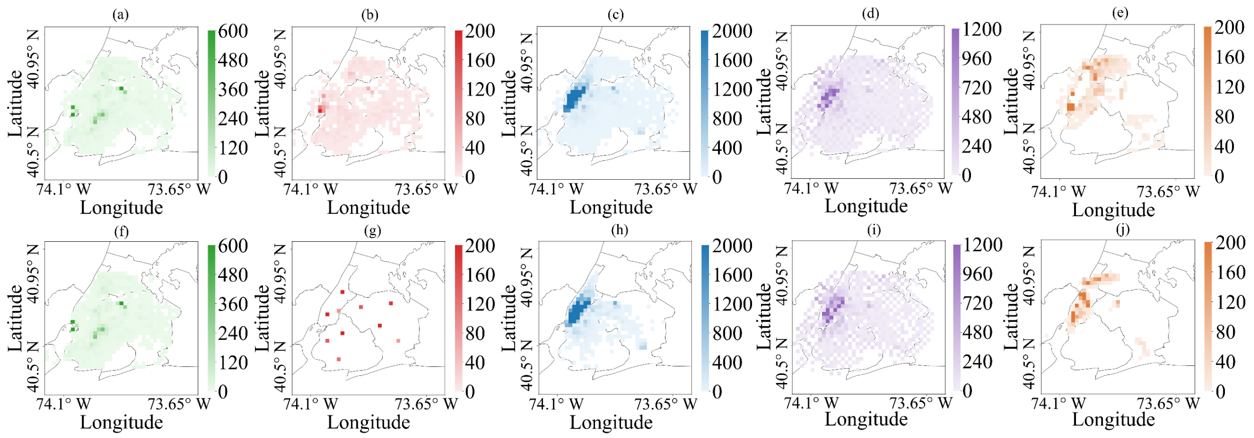

Figure 4.3. Distributions of dispatch originations and destinations for different dispatch methods.

As explained in Section 3.3, the curbside parking slots in NYC are widely distributed over the entire city, and vehicles can easily find a nearby parking slot without driving too far, which explains why Nearest-All dispatch originations are similar with Nearest-Cluster dispatch destinations. The pattern of dispatch origination distribution is similar with Nearest-All, but the dispatch destination distributions are different. Since I use scattered parking depots as dispatch destinations rather than the widely spread curbside parking slots as in Nearest-All, the idle vehicles headed towards the selected depots as expected. The dispatch destinations of Hotspot are randomly sampled from the same NYC taxi demands throughout the entire day. As result, the density distributions of dispatch destinations are highly similar over the four periods, but the number of dispatch trips varies over the four periods as the number of idle vehicles is different. The distribution of dispatch destinations is highly concentrated in Manhattan compared with Nearest-All and Nearest-Cluster, which implies that the idle vehicle supply in Manhattan should be higher for Hotspot. I show this pattern in Section 4.3. Compared with Hotspot, the dispatch destinations are less focused on Manhattan and are more spread-out over the entire NYC. The reason is that in the case of NYC, trip demand is heavily concentrated in Manhattan, which means that Manhattan will mostly absorb idle vehicles from surrounding regions according to the rule of R1 and R2. With the locally search restriction and the awareness of rider-vehicle imbalance, Hierarchical-Fill should prioritize shorter and fewer dispatch trips than Hotspot, where idle vehicles mostly move towards Manhattan regardless of how far they are from Manhattan and how many vehicles are already in Manhattan. This is consistent with the actual observation from Table 4.1 that Hierarchical-Fill dispatch about 40% less frequently and 24% shorter trips than Hotspot. Similar with Hotspot and Hierarchical-Fill, DQN heavily focuses on moving vehicles to high-demand regions (mainly Manhattan and JFK region) as well. However, DQN only chooses to relocate surrounding vehicles that are close enough (Manhattan borders, like Bronx and north Brooklyn). For idle vehicles in low-demand regions, DQN does not even rebalance them possibly due to the low expected matching possibility of any dispatch action. In addition to the similar features of local dispatch restriction and rider-vehicle distribution awareness as in Hierarchical-Fill, DQN takes the cost of dispatch trips into consideration and further optimizes vehicle dispatching rather than using the rules as in Hierarchical-Fill. With the better problem formulation than Hierarchical-Fill, DQN dispatch about 90% less trips than Hierarchical-Fill.

4.1.2 Vehicle Pick-up

As explained in Section 4.1.1, the vehicle performance of Nearest-All is highly similar with No-Dispatch as well for pick-up component. The mean pick-up distance of Nearest-Cluster is longer than No-Dispatch and Nearest-All. Given the observation that the dispatch distance of Nearest-Cluster is about three times higher than Nearest-All as explained in Section 4.1.1, Nearest-Cluster in this study is even worse than Nearest-All and No-Dispatch. Considering the dispatch origination and destination patterns of Nearest-Cluster in conjunction with Hotspot, Hierarchical-Fill, and DQN, I suspect that this is related to the selection of parking depots. While Hotspot, Hierarchical-Fill, and DQN all involve moving idle vehicles to Manhattan, Nearest-Cluster navigate some idle vehicles around Manhattan to parking depots in Manhattan outskirts as depicted in Figure 4.3. I also show that the idle vehicle supply of Nearest-Cluster is indeed even lower than Nearest-All in Section 4.3. Hotspot and Hierarchical-Fill are observed to have shortest mean pick-up distance, and DQN has a mean pick-up distance between the proactive methods and the conservative methods. This gap might be related to vehicle dispatch distance and dispatch method as explained in Section 4.1.1. I also observed that rider service rate is negatively correlated with vehicle pick-up time, and I discuss this pattern in Section 4.3. I illustrate the probability density distribution of pick-up trips' distance for each method in Figure 4.4. The pick-up trips of Hotspot and Hierarchical-Fill are similarly distributed and shortest among all methods. While the pick-up trips of No-Dispatch, Nearest-All, and Nearest-Cluster all are similarly distributed and longest among all methods, DQN is also similarly distributed with them but has a smaller mean value.

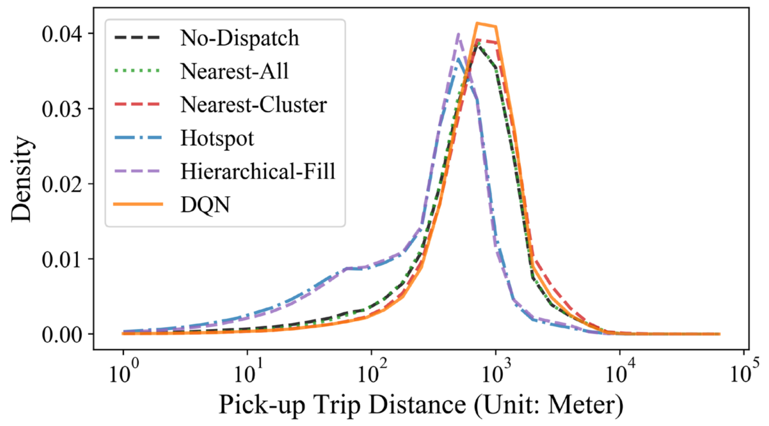

Figure 4.4. Probability density distribution of pick-up trip distance.

4.1.3 Vehicle Service

I illustrate the probability density distribution of service trips' distance for each method in Figure 4.5. The distributions of service trip distance are highly similar for all methods, which means that none of the dispatch methods prioritizes to serve trips of a certain length. As result, the fleet total service distance is linearly dependent on the number of served riders as shown in Table 4.1.

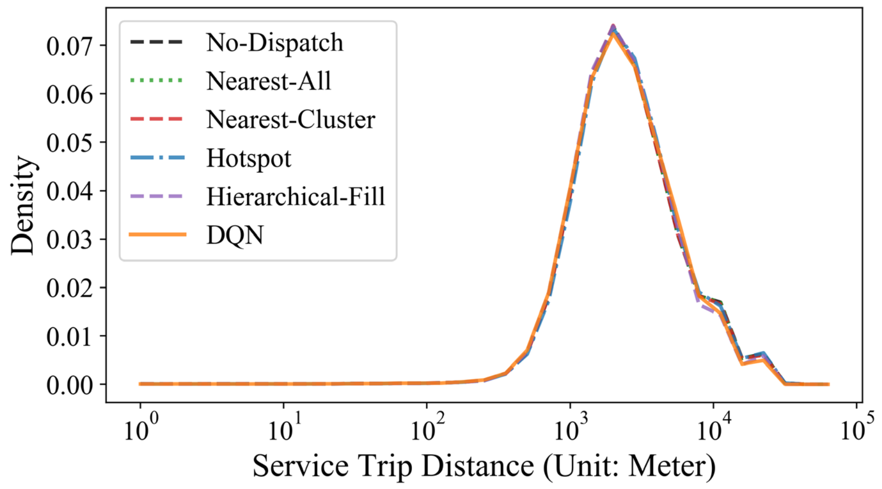

Figure 4.5. Probability density distribution of service trip distance.

4.2 Rider Service Level

I define rider service rate as the proportion of served riders out of all riders entered the system and rider waiting time as the duration a served rider spent waiting to be matched with an idle vehicle plus the duration the matched vehicle spent picking up the rider. Similar with vehicle performance, I compare the rider service level of the benchmarks and DQN in Table 4.2.

Table 4.2. Rider service level comparison.

Dispatch Method	Number of Served Riders	Service Rate (%)	Mean Waiting Time (unit: minute)
No-Dispatch	343,410	67.17	3.94
Nearest-All	344,602	67.40	3.97
Nearest-Cluster	309,498	60.54	4.56
Hotspot	418,380	81.83	2.02
Hierarchical-Fill	411,569	80.50	2.13
DQN	400,101	78.26	3.01

As explained in Section 4.1.1, the rider service level of Nearest-All is also highly similar with No-Dispatch. For Nearest-Cluster, the rider service level is worse than No-Dispatch and Nearest-All. Despite the longer fleet total dispatch distance than No-Dispatch and Nearest-All, the dispatch trips of Nearest-Cluster negatively contributed to the fleet, and the method underserved the trip demands than the case even without the dispatch trips. By contrast, Hotspot and Hierarchical-Fill achieved the best rider service level with the extra dispatch trips, suggesting that the dispatch trips contributed positively to the fleet. DQN, with slightly longer fleet dispatch distance than conservative methods but much shorter than proactive methods, achieved similar rider service level with proactive methods, which suggests that the positive contribution of DQN dispatch trips are more efficient than Hotspot and Hierarchical-Fill.

Moreover, the spatial difference in rider service rate is shown in Figure 4.6. I set No-Dispatch as the baseline and show the grid-level rider service rate ratio of other dispatch methods compared to the baseline values. Note that Figure 4.6a-e correspond to Nearest-All, Nearest-Cluster, Hotspot, Hierarchical-Fill, and DQN methods, respectively. As explained in Section 4.1.1, the spatial pattern of rider service rate of Nearest-All is highly similar with No-Dispatch as expected, and Nearest-Cluster is observed to suffer a system-wide underservice in the entire NYC region due to poorly selected parking depot. For high-demand regions like Manhattan and JFK area, Hotspot, Hierarchical-Fill, and DQN are observed to have better service rate than baseline owing to the dispatch focus as depicted in Figure 4.3. For low-demand regions, the Hotspot method, without awareness of rider and vehicle distribution nor constraints on dispatch trips, is observed to suffer a severe underservice as a result of overly pulling way idle vehicles in low demand regions. By contrast, the Hierarchical-Fill method, with a local search constraint (R1 and R2) and a proactive vehicle spread-out mechanism (R3 and R4), focused on high-demand regions less than the Hotspot method and achieved decent service rate in low demand regions. With the definition of dispatch benefit and cost and the optimization framework, DQN also achieved better service rate on high-demand regions. However, DQN did not proactively promote dispatch in low-demand regions like Hierarchical-Fill as shown in Figure 4.3. DQN maintained some idle vehicles in these regions by not dispatching them, which helped alleviating the severe underservice encountered by the Hotspot method but not as effective as Hierarchical-Fill.

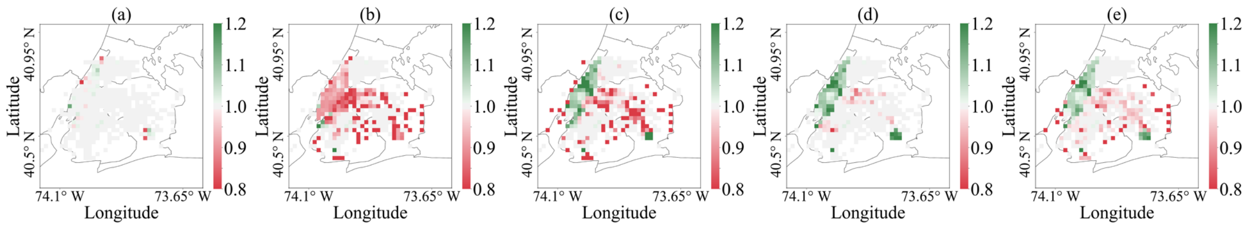

Figure 4.6. Grid-level rider service rate ratio.

Moreover, I quantify and compare the dispatch efficiency of these methods in Figure 4.7. In addition, I notice that rider service rate is negatively correlated with mean rider waiting time, which suggests that the methods with better rider service level not only identified more riders but also located vehicles closer to the served riders.

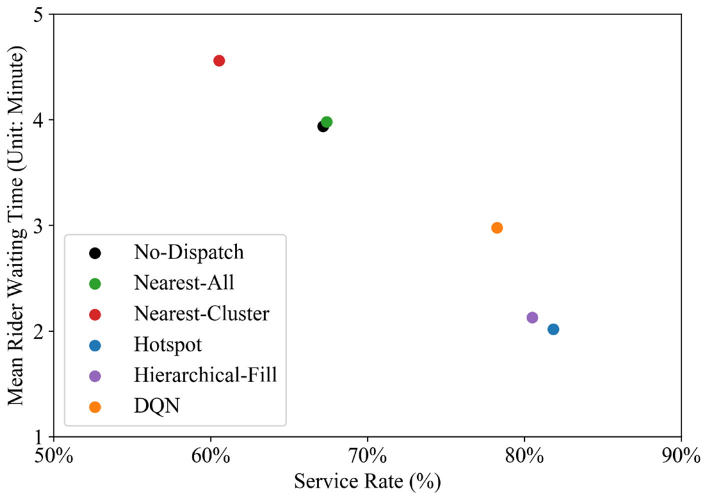

Figure 4.7. Mean rider waiting time versus service rate.

4.3 System-Level Comparison

To quantify the difference in fleet utilization caused by different dispatch methods, I further breakdown the different components of fleet traveled distance in Figure 4.8. It is observed that the fleet service and pick-up components are similar for all dispatch methods and the major difference is the dispatch components. Specifically, Hotspot and Hierarchical-Fill have observably higher dispatch components than the rest dispatch methods because of their active dispatch strategies.

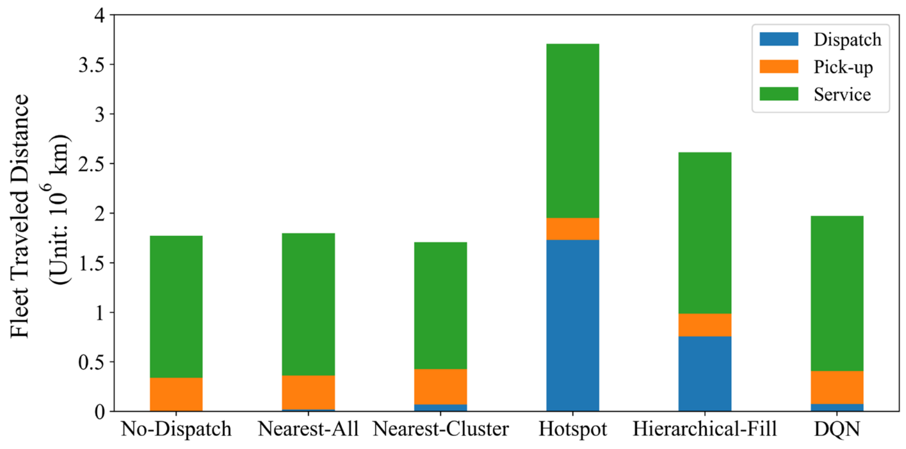

Figure 4.8. Components of fleet traveled distance.

Moreover, to reflect the relationship between vehicle in-operation time and rider service rate, I compare the average in-operation proportion versus rider service rate for all dispatch methods in Figure 4.9. It is observed that the average in-operation proportion is positively correlated with and mostly linear to rider service rate, which means that different dispatch methods served riders with similar service time and vehicle in-operation time is mainly caused by the number of served riders rather than the service time of each rider.

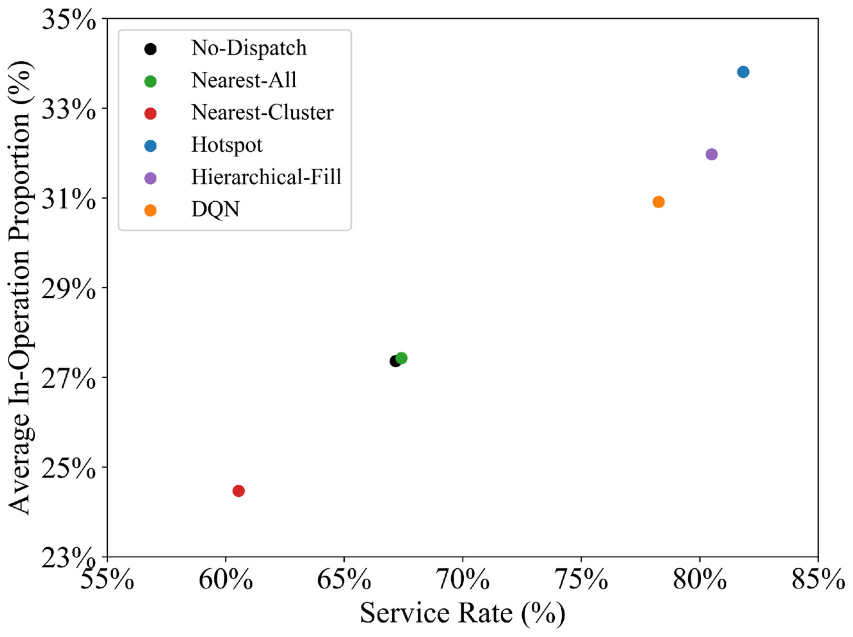

Figure 4.9. Average vehicle occupancy versus service rate.

In addition, I notice that the number of served riders in each hour are different for different dispatch methods as shown in Figure 4.10. It is observed that the rider service rate advantages of Hotspot, Hierarchical-Fill, and DQN are more distinguishable to the other dispatch methods during morning peak and evening peak owing to the dispatch focus as explained in Section 4.1.1.

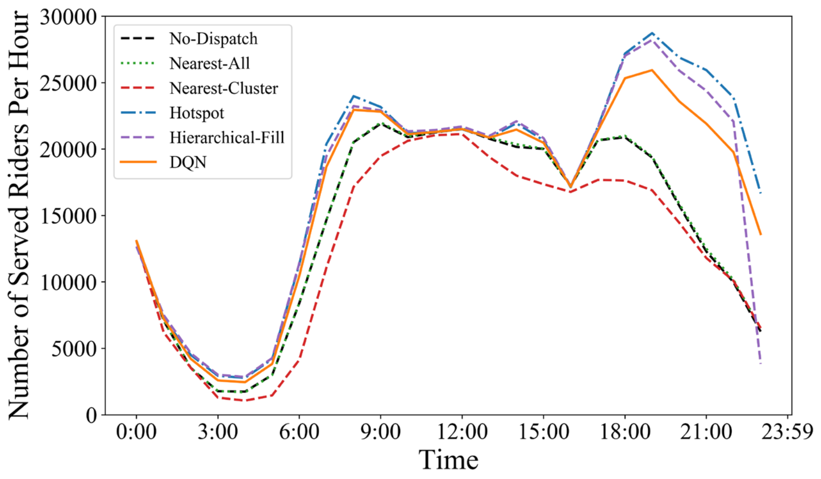

Figure 4.10. Number of served riders per hour.

Furthermore, to reflect the idle vehicle distribution of different dispatch methods, I compare the number of idle vehicles in Manhattan for different dispatch methods during the evaluation day in Figure 4.11. The results show that Hotspot, Hierarchical-Fill, and DQN expectedly maintained more idle vehicles in Manhattan than other dispatch methods.

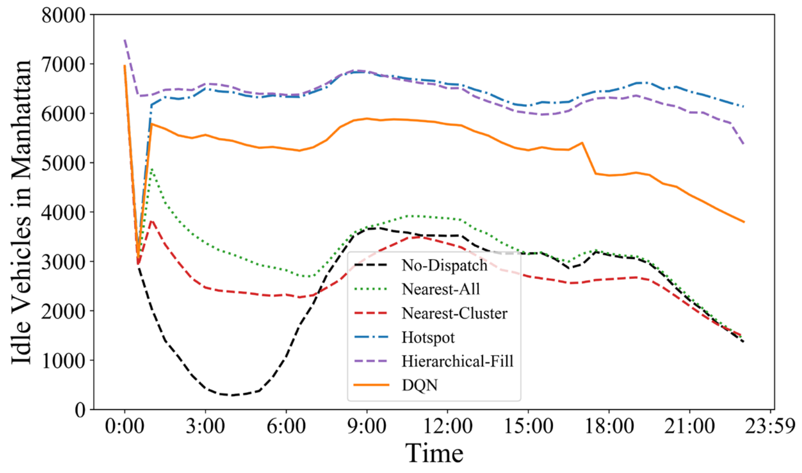

Figure 4.11. Number of idle vehicles in Manhattan.

Lastly, to quantify and compare the net benefit brought by the dispatch trips of different dispatch methods to the fleet system, I compare the dispatch efficiency among dispatch methods as in Figure 4.12. With fewer served riders than No-Dispatch, Nearest-Cluster is negative in fleet dispatch efficiency as expected. Nearest-All achieved slightly positive fleet dispatch efficiency (around 0.1 extra rider/meter) with dispatch trips to nearest parking lots, which suggests that the minor extra dispatch distance is contributing positively and not very effectively. Despite that Hotspot and Hierarchical-Fill served more riders, their fleet dispatch efficiencies are actually similar with Nearest-All when averaged over the extremely long dispatch distances. By contrast, the fleet dispatch efficiency of DQN (around 0.75 extra rider/meter) is about seven times higher than other methods with positive dispatch efficiency, which suggests that DQN is the most efficient dispatch method of all the approaches.

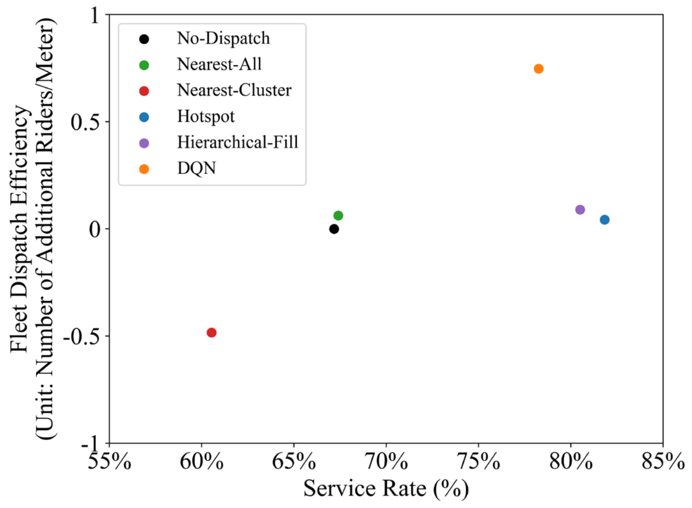

Figure 4.12. Fleet dispatch efficiency versus service rate.

5. CONCLUSION AND FUTURE WORK

In summary, I integrated a reinforcement learning-based algorithm (DQN) into an agent-based autonomous taxi fleet model and evaluated the DQN algorithm using a real-world scale simulation. I compared the performance of DQN with common dispatch methods in related literatures. While DQN slightly underserved the fleet comparing with the best serving method, DQN is much more efficient than all of them considering the cost of dispatching. The benefit of DQN also lies in its distributed nature, which scales well with the size of dispatch problem.

Multiple aspects of this study can be improved in future research. To address the service rate deficiency, a better spatiotemporal global coordination (e.g., multi-step and long-distance dispatch) and a better rewarding mechanism are needed. In addition, the rider service level itself can be reflected in the reward function definition in terms of some variables. It would be more realistic to run continually for multiple days without the vehicle initialization at the beginning of each day. Moreover, this simulator can be easily generalized to a ride-sharing scenario and a station-based electric vehicle scenario. It would be interesting to migrate this DQN framework to these similar operational problems, but more sophisticated s_t, a_t, r_t definitions need to be designed. A congestible road network can be included into the simulator as well to consider the impact of dispatch trips on congestion. Some other influencing factors, such as trip fare pricing, long-term changes of rider-system interactions can be introduced into the decision-making of the system as well, and their joint effect can be further explored by more sophisticated experiments.

REFERENCES

Al-Abbasi, A.O., Ghosh, A., Aggarwal, V., 2019. DeepPool: Distributed Model-Free Algorithm for Ride-Sharing Using Deep Reinforcement Learning. IEEE Trans. Intell. Transp. Syst. https://doi.org/10.1109/TITS.2019.2931830

Boesch, P.M., Ciari, F., Axhausen, K.W., 2016. Autonomous vehicle fleet sizes required to serve different levels of demand. Transp. Res. Rec. https://doi.org/10.3141/2542-13

Dandl, F., Hyland, M., Bogenberger, K., Mahmassani, H.S., 2019. Evaluating the impact of spatio-temporal demand forecast aggregation on the operational performance of shared autonomous mobility fleets. Transportation (Amst). https://doi.org/10.1007/s11116-019-10007-9

De Souza, F., Gurumurthy, K.M., Auld, J., Kockelman, K.M., 2020. A Repositioning Method for Shared Autonomous Vehicles Operation, in: Procedia Computer Science. https://doi.org/10.1016/j.procs.2020.03.154

Fagnant, D.J., 2016. Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in Austin, Texas. Transportation (Amst).

Fagnant, D.J., Kockelman, K.M., 2014. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. Transp. Res. Part C Emerg. Technol. https://doi.org/10.1016/j.trc.2013.12.001

Haliem, M., Mani, G., Aggarwal, V., Bhargava, B., 2020. A Distributed Model-Free Ride-Sharing Approach for Joint Matching, Pricing, and Dispatching using Deep Reinforcement Learning 1–15. https://doi.org/10.1145/3408308.3431114

Heilig, M., Hilgert, T., Mallig, N., Kagerbauer, M., Vortisch, P., 2017. Potentials of Autonomous Vehicles in a Changing Private Transportation System - A Case Study in the Stuttgart Region, in: Transportation Research Procedia. https://doi.org/10.1016/j.trpro.2017.07.004

Holler, J., Vuorio, R., Qin, Z., Tang, X., Jiao, Y., Jin, T., Singh, S., Wang, C., Ye, J., 2019. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem, in: Proceedings - IEEE International Conference on Data Mining, ICDM. https://doi.org/10.1109/ICDM.2019.00129

Huang, Y., Kockelman, K.M., Garikapati, V., 2020. Use of shared automated vehicles for first-mile last-mile service: micro-simulation of rail-transit connections in Austin, Texas. 99th Annu. Meet. Transp. Res. Board.

Iglesias, R., Rossi, F., Wang, K., Hallac, D., Leskovec, J., Pavone, M., 2018. Data-driven model predictive control of autonomous mobility-on-demand systems, in: Proceedings - IEEE International Conference on Robotics and Automation. https://doi.org/10.1109/ICRA.2018.8460966

Insurance Institute for Highway Safety, 2020. Autonomous vehicle laws [WWW Document]. URL https://www.iihs.org/topics/advanced-driver-assistance/autonomous-vehicle-laws (accessed 10.13.20).

Javanshour, F., Dia, H., Duncan, G., 2019. Exploring the performance of autonomous mobility on-demand systems under demand uncertainty. Transp. A Transp. Sci. https://doi.org/10.1080/23249935.2018.1528485

Korte, B., Vygen, J., 2008. Kombinatorische Optimierung: Theorie und Algorithmen. Ann. Phys. (N. Y).

Levin, M., Li, T., Boyles, S., Kockelman, K., 2016. A general framework for modeling shared autonomous vehicles. 95th Annu. Meet. Transp. Res. Board.

Lokhandwala, M., Cai, H., 2018. Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of NYC. Transp. Res. Part C Emerg. Technol. https://doi.org/10.1016/j.trc.2018.10.007

Maciejewski, M., Bischoff, J., 2018. Congestion effects of autonomous taxi fleets. Transport 33. https://doi.org/10.3846/16484142.2017.1347827

Manchella, K., Umrawal, A.K., Aggarwal, V., 2020. FlexPool: A Distributed Model-Free Deep Reinforcement Learning Algorithm for Joint Passengers & Goods Transportation. arXiv.

Martinez, L.M., Correia, G.H.A., Viegas, J.M., 2015. An agent-based simulation model to assess the impacts of introducing a shared-taxi system: An application to Lisbon (Portugal). J. Adv. Transp. https://doi.org/10.1002/atr.1283

Martinez, L.M., Viegas, J.M., 2017. Assessing the impacts of deploying a shared self-driving urban mobility system: An agent-based model applied to the city of Lisbon, Portugal. Int. J. Transp. Sci. Technol. https://doi.org/10.1016/j.ijtst.2017.05.005

NYC DOT, 2020. NYC Taxi and Limousine Commission - Trip Record Data [WWW Document]. URL https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page (accessed 7.30.20).

NYC Open Data, 2016. Parking Lot | NYC Open Data [WWW Document]. URL https://data.cityofnewyork.us/City-Government/Parking-Lot/h7zy-iq3d

Oda, T., Joe-Wong, C., 2018. MOVI: A Model-Free Approach to Dynamic Fleet Management, in: Proceedings - IEEE INFOCOM. https://doi.org/10.1109/INFOCOM.2018.8485988

Pavone, M., Smith, S.L., Frazzoli, E., Rus, D., 2012. Robotic load balancing for mobility-on-demand systems. Int. J. Rob. Res. https://doi.org/10.1177/0278364912444766

Shen, Y., Zhang, H., Zhao, J., 2018. Integrating shared autonomous vehicle in public transportation system: A supply-side simulation of the first-mile service in Singapore. Transp. Res. Part A Policy Pract. 113. https://doi.org/10.1016/j.tra.2018.04.004

Spieser, K., Samaranayake, S., Frazzoli, E., 2016. Vehicle routing for shared-mobility systems with time-varying demand, in: Proceedings of the American Control Conference. https://doi.org/10.1109/ACC.2016.7525011

Spieser, K., Treleaven, K., Zhang, R., Frazzoli, E., Morton, D., Pavone, M., 2014. Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems: A Case Study in Singapore. https://doi.org/10.1007/978-3-319-05990-7_20

Sutton, R.S., Barto, A.G., 2012. Reinforcement learning: An Introduction Second edition. Learning. https://doi.org/10.1109/MED.2013.6608833

Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double Q-Learning, in: 30th AAAI Conference on Artificial Intelligence, AAAI 2016.

Wallar, A., Van Der Zee, M., Alonso-Mora, J., Rus, D., 2018. Vehicle Rebalancing for Mobility-on-Demand Systems with Ride-Sharing, in: IEEE International Conference on Intelligent Robots and Systems. https://doi.org/10.1109/IROS.2018.8593743

Wen, J., Zhao, J., Jaillet, P., 2018. Rebalancing shared mobility-on-demand systems: A reinforcement learning approach, in: IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC. https://doi.org/10.1109/ITSC.2017.8317908

Yan, H., Kockelman, K.M., Gurumurthy, K.M., 2020. Shared autonomous vehicle fleet performance: Impacts of trip densities and parking limitations. Transp. Res. Part D Transp. Environ. 89. https://doi.org/10.1016/j.trd.2020.102577

Zhang, R., Pavone, M., 2016. Control of robotic mobility-on-demand systems: A queueing-theoretical perspective. Int. J. Rob. Res. https://doi.org/10.1177/0278364915581863

APPENDIX A. SUPPLEMENTARY FIGURES.

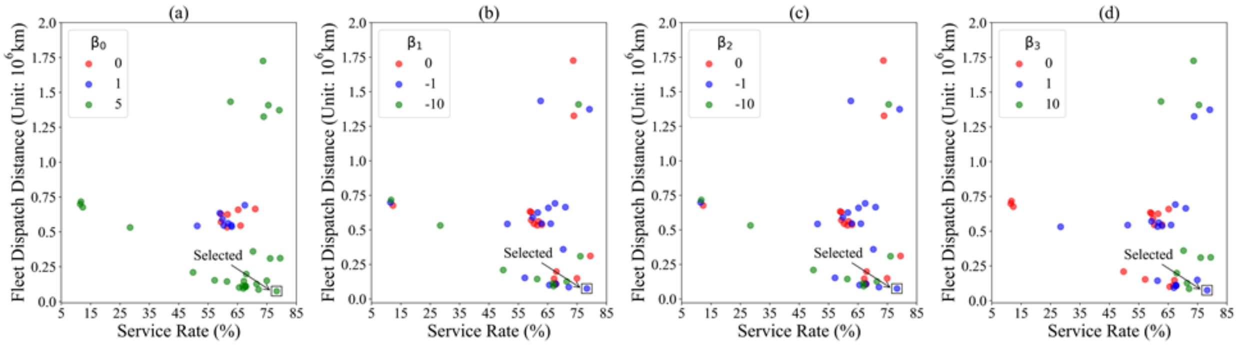

Figure A.1. Scatter plot of fleet dispatch distance versus service rate for all tested and selected DQN β hyperparameters. (a)-(d) show the patterns for β_0, β_1, β_2, and β_3 respectively.

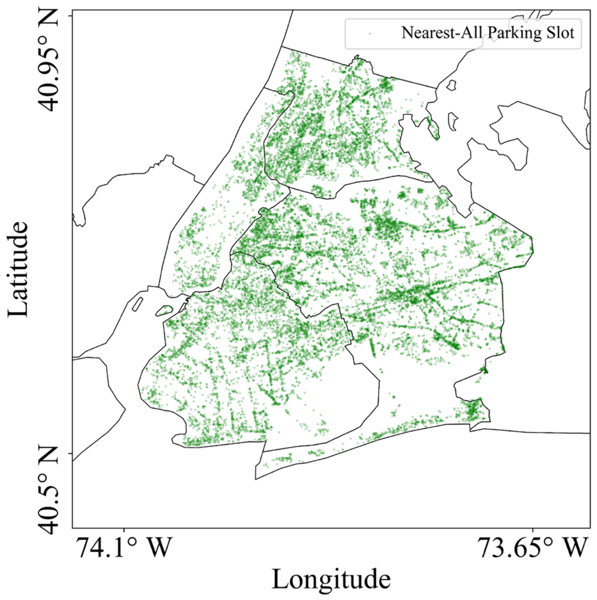

Figure A.2. Spatial distribution of Nearest-All parking slot.

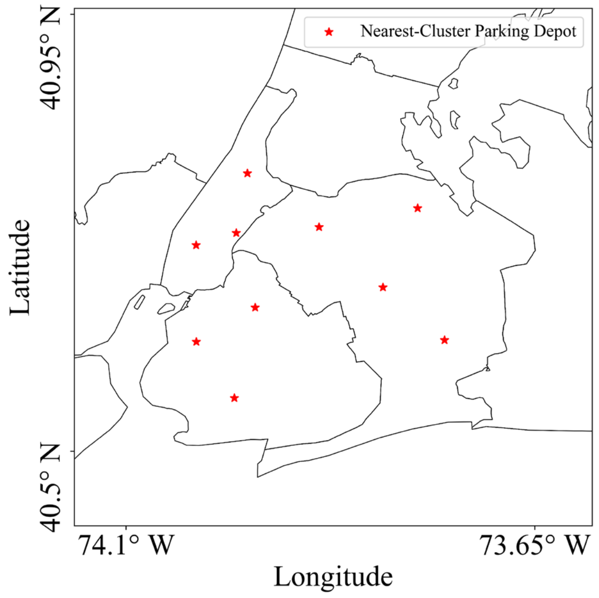

Figure A.3. Spatial distribution of Nearest-Cluster parking depot.

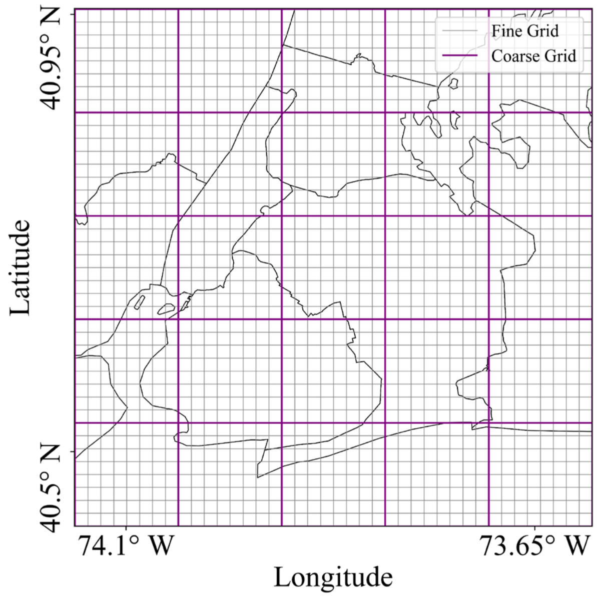

Figure A.4. Coarser and finer grid systems of Hierarchical-Fill.

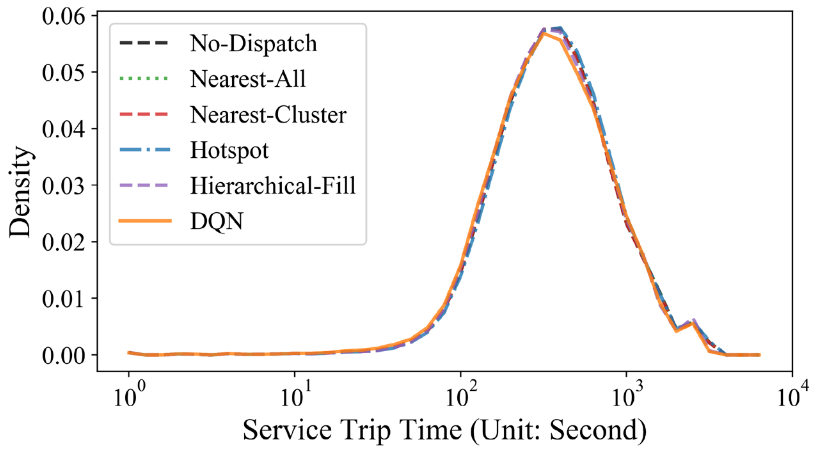

Figure A.5. Probability density distribution of service trip time.

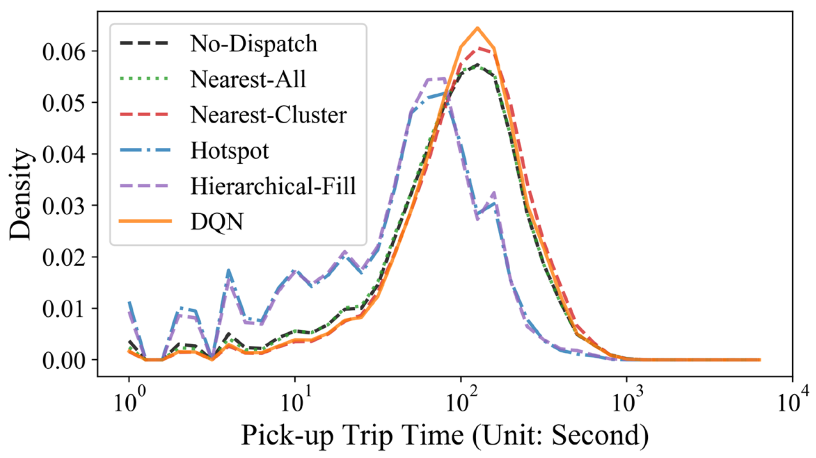

Figure A.6. Probability density distribution of pick-up trip time.

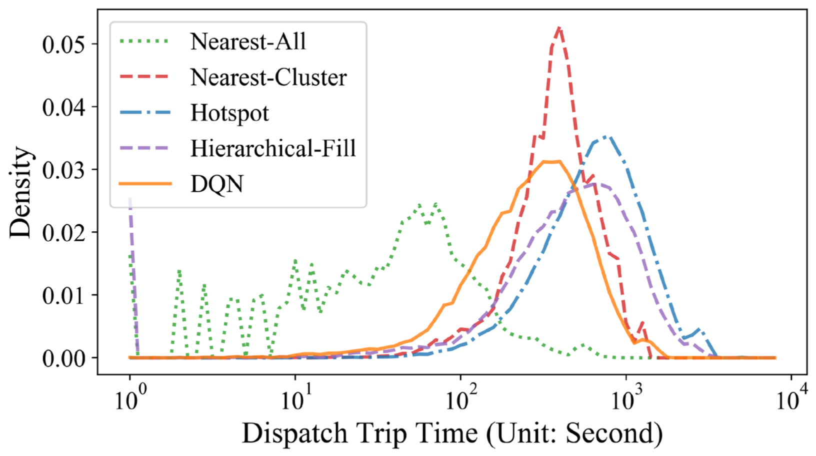

Figure A.7. Probability density distribution of dispatch trip time.

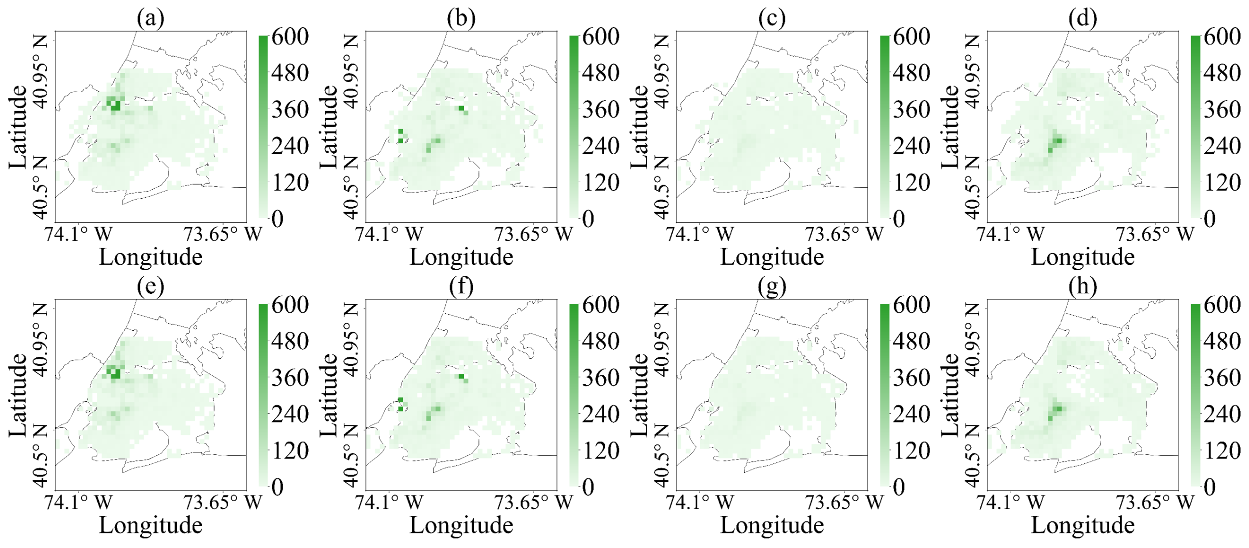

Figure A.8. Distribution of dispatch origination and destination for Nearest-All. (a)-(d) show the dispatch origination distributions for early morning, morning peak, afternoon, and evening peak. Similarly, (e)-(h) show the dispatch destination distributions for the corresponding four periods.

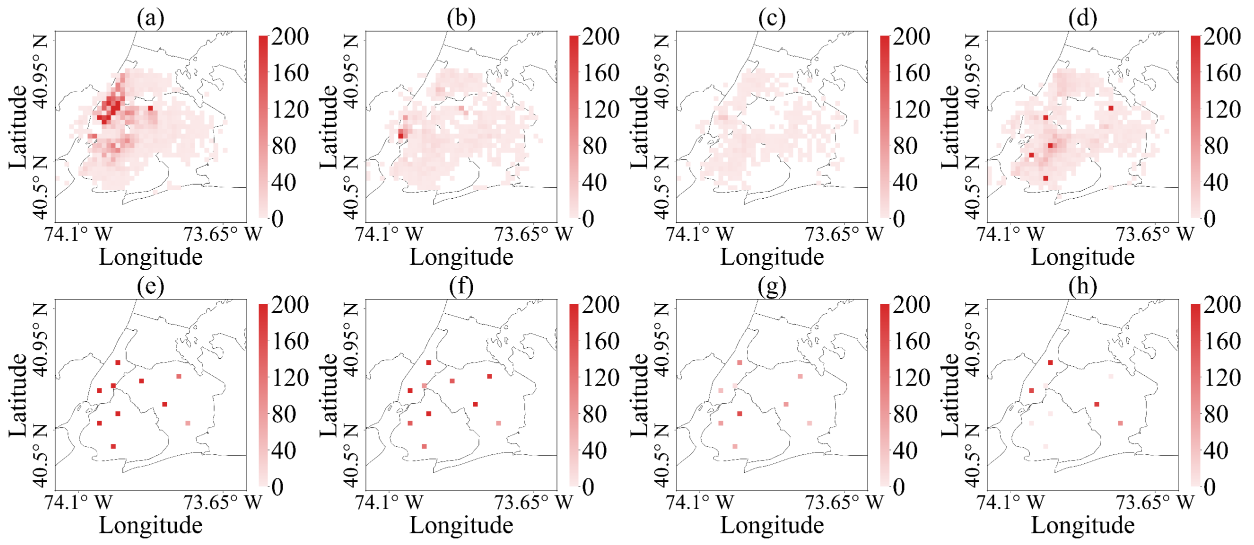
Figure A.9. Distribution of dispatch origination and destination for Nearest-Cluster. (a)-(d) show the dispatch origination distributions for early morning, morning peak, afternoon, and evening peak. Similarly, (e)-(h) show the dispatch destination distributions for the corresponding four periods.

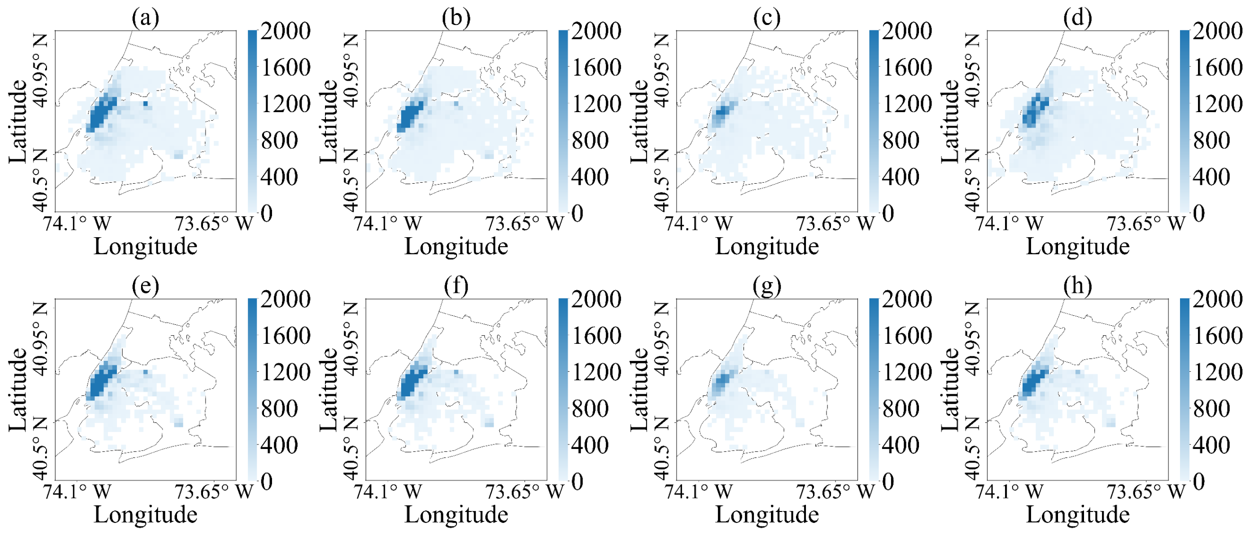
Figure A.10. Distribution of dispatch origination and destination for Hotspot. (a)-(d) show the dispatch origination distributions for early morning, morning peak, afternoon, and evening peak. Similarly, (e)-(h) show the dispatch destination distributions for the corresponding four periods.

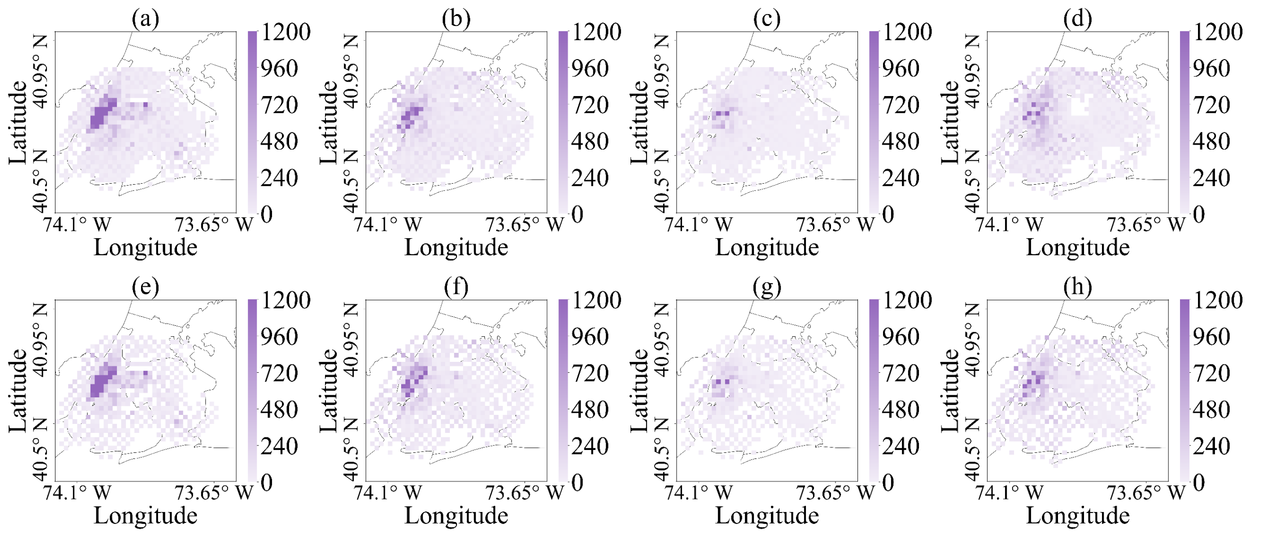

Figure A.11. Distribution of dispatch origination and destination for Hierarchical-Fill. (a)-(d) show the dispatch origination distributions for early morning, morning peak, afternoon, and evening peak. Similarly, (e)-(h) show the dispatch destination distributions for the corresponding four periods.

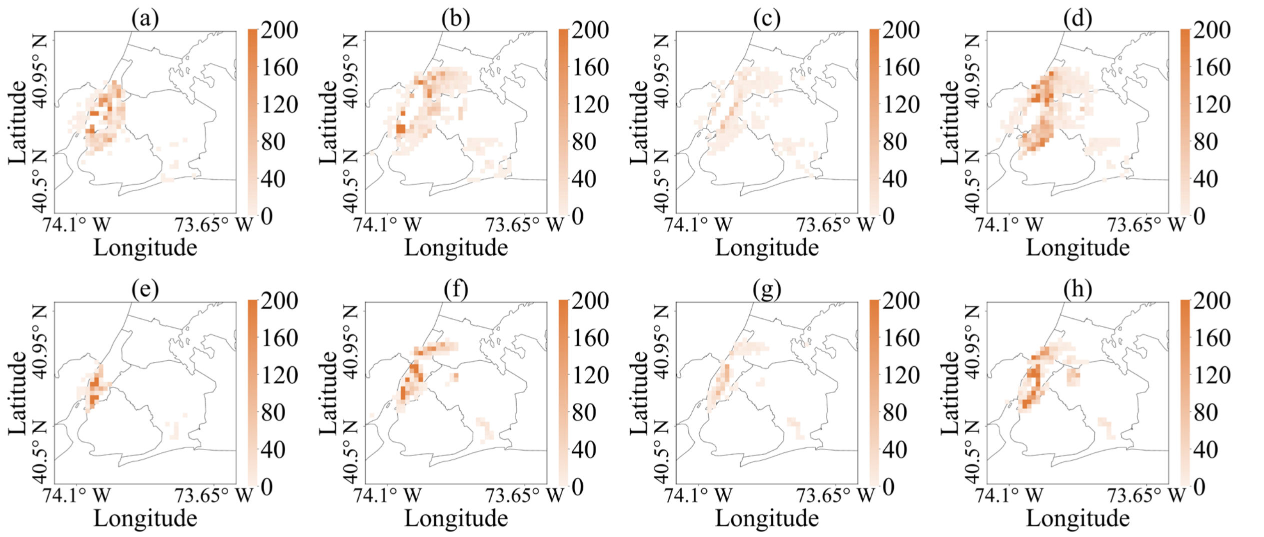

Figure A.12. Distribution of dispatch origination and destination for DQN. (a)-(d) show the dispatch origination distributions for early morning, morning peak, afternoon, and evening peak. Similarly, (e)-(h) show the dispatch destination distributions for the corresponding four periods.