TLab: 1st Place Solution in the Development Phase of Vehicle Repositioning Task

Fanyou Wu
wu1297@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Yang Liu
230179629@seu.edu.cn
Southeast University
Nanjing, Jiangsu, China

ABSTRACT

The Vehicle dispatching system has always been one of the most critical problems in online taxi-hailing platforms to adapt the operation and management strategy to the dynamics in demand and supply. In this paper, we propose a single agent deep reinforcement learning approach for vehicle repositioning by deploying idle vehicles to specific locations to anticipate future demand at the destination. A global pruned action space, which encompasses a set of M discrete actions, is used in this study. It can benefit drivers by avoiding traveling to distant outskirts where there are few order requests. We include not only the local information of vehicles as the state, but also the global information in the whole study area, i.e., the real-time distribution of orders, vehicles, and rewards. For better performance, we design a simulator using the Julia programming language, which brings about over ten times optimization in speed compared with the simulator implementation in Python. An efficient simulator, along with the tailored visualization tool, helps us with the rapid update of our framework. Concerning the performance, our team (TLab) has ranked first place in the development phase of the vehicle repositioning task of KDD Cup 2020.

CCS CONCEPTS

• **General and reference** → **General literature**; • **Computing methodologies** → **Intelligent agents**.

1 INTRODUCTION

In the past decade, the online taxi-hailing service has gained extensive attention from urban travelers. The innovative operating strategies offer exceptional convenience and flexibility to user experience, making it an indispensable component of the multi-modal urban transportation network. Maintaining a balance between the increasing demand and limited supply is the crux of the online taxi-hailing platform operation.

Better knowledge about future demand is useful in lowering the vehicle vacancy ratio and the operation cost. Abundant research relating to demand prediction focused on digging the spatio-temporal correlations of traffic dynamics. A convolutional neural network (CNN) is often employed to model spatial relationships. Liu et al. [6] designed a novel ensemble learning framework for taxi demand prediction, which contains three attention blocks modeling the relationships among base models, among spatial locations and among positions of augmented feature maps. Inspired by 'personalization' in recommendation systems [7], Liu et al. [8] proposed a personalized model for large-scale online taxi-hailing demand prediction with two attention blocks to capture both spatial and temporal perspective. Some researchers argued that the road network could be better modeled using a graph rather than a grid. In addition to the spatial and temporal information captured by a local CNN and LSTM, Yao et al. [17] constructed a location graph utilizing the land use data to represent the similarity among regions. Lin et al. [5] proposed the use of a data-driven graph filter to learn the hidden heterogeneous correlations between stations when applying GCN to the bike demand prediction. Although graph-based models are powerful in terms of predictive ability, the major hindrance of their applications in real scenarios is that updated base maps are not always available. Liu et al. [9] use fine-grained partitioning to maintain the road network topology and perform traffic prediction using the industrial-scale real world probe data in three cities, covering approximately six thousand square kilometers in total.

Apart from predicting future demand, achieving the balance between demand and supply requires reasonable strategies for vehicle repositioning and fleet management. The efficiency of the overall platform can be enhanced by reallocating vacant vehicles to regions with a large demand gap in advance. Existing researches in this field concentrate on mathematical programming. Song and Earl [15] focused on determining the optimal strategies for vehicle repositioning and the size of the fleet. Their model minimizes the sum of the maintenance cost, repositioning cost, and leasing cost, incorporating the uncertainty in vehicle arrival and repositioning time. Nair and Miller-Hooks [12] designed a stochastic mixed integer program with joint chance constraints to generate a plan of vehicle repositioning at the lowest cost. Roy et al. [14] proposed models based on the queueing theory to study various repositioning strategies, the optimization objective of which is to minimize the cost of the platform. They also designed a two-phase algorithm to

solve the nonlinear programming problem with non-differentiable objectives and interrelated decision variables. He et al. [1] formulated vehicle repositioning problem as a stochastic dynamic program, where distributionally robust optimization is used to allow for the temporal dependence of demand. Ma et al. [10] considered the integration of on-demand mobility services and public transit systems and developed an effective dispatching algorithm to optimize this novel integrated strategy.

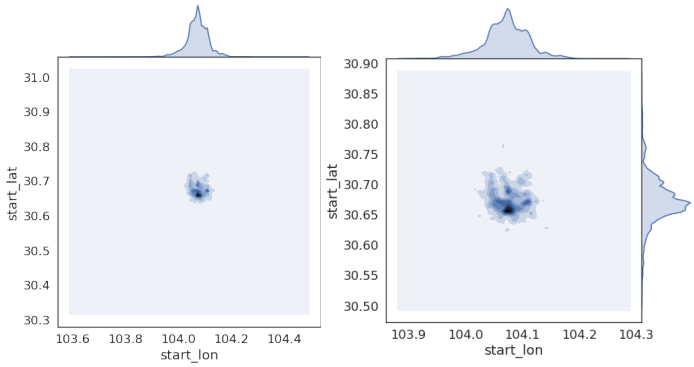
It should be aware that the demand and supply are interdependent, which always changes when a repositioning strategy is applied. However, traditional mathematical programming and supervised learning techniques are difficult to capture these dynamics. Therefore, deep reinforcement learning (DRL) is often adopted to address the complex interactions between demand and supply in vehicle repositioning problems. Lin et al. [4] proposed a contextual fleet management framework based on multi-agent reinforcement learning, comprised of two modules, i.e., contextual deep Q-learning and contextual multi-agent actor-critic. Mao et al. [11] also developed a model-free method, i.e., a policy-based DRL framework, based on the actor-critic algorithm for vehicle dispatching. In addition to drivers, a central management system, which makes decisions for vehicle dispatching, is considered as an agent in Holler et al. [2]'s formulation. Their network is also capable of learning a global embedding of orders and drivers. Contrary to the central management system, Zhou et al. [18] proposed a decentralized strategy based on DRL, which does not require communication between agents, to adapt to large-scale vehicle repositioning problems. Jin et al. [3] modeled vehicle repositioning as a large-scale parallel ranking problem. A geographical hierarchy is constructed to accommodate the agent coordinations in various regions, and a multi-head attention mechanism is used for the hierarchical decision-making process.

2 DATA DESCRIPTION

The data used in this paper for the simulation environment are extracted from the online car-hailing record of Chengdu, China, which contains a large number of the trip trajectory (more than forty million trajectory records per day) and order data (more than seven million order records in total) in November 2016. GPS trajectory records contain anonymized driver id, user id, timestamp, and the corresponding latitude and longitude. Each trip order record comprises the information of anonymous order id, starting time, ending time, starting position, ending position, and reward of the order. The spatial distribution of the raw order data is demonstrated in Figure 1a . Considering that the model may fail to converge due to data sparsity, we narrow down the study area. As shown in Figure 1b, the longitudinal and latitudinal ranges of the selected study area are $[103.89E, 104.28E]$ and $[30.5N, 30.88N]$, respectively. Approximately 99.67% of the orders in the 30-day dataset fall in our selected study area.

The statistics of hourly orders in hexagonal grids are listed in Table 1. In the dataset, 50% of hexagonal grids receive 0.3 orders, which further indicates the severity of data sparsity.

The distribution of hourly orders is demonstrated in Figure 2. It can be observed that the demand is low before 7 a.m., while most demand distributes in the other 17 hours. Different from passenger



(a) Full Area **(b) Study Area**

Figure 1: Spatial distribution of orders.

Table 1: Statistics of hourly orders

	Hourly orders (order per hour, oph)
mean	6.308907
std	21.096421
min	0.001389
25%	0.069444
50%	0.313194
75%	2.179167
max	315.384722

demand of metro and bus, neither morning peak nor evening peak can be observed concerning online taxi-hailing services.

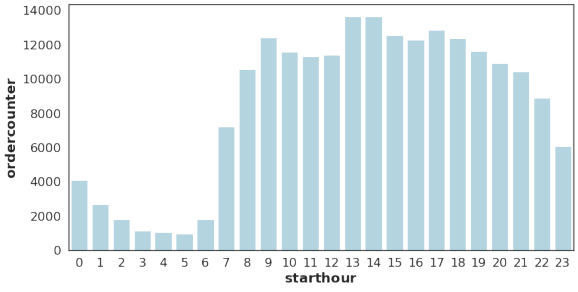


Figure 2: Hourly order distribution

Table 2 lists the statistics of the driver's operation time (with passengers in the vehicle) and online time on November 1. If no order request is received within 30 minutes, the driver will be considered offline, and thereafter the time will not include in the online time.

3 PROBLEM STATEMENT

The goal of vehicle repositioning task is to maximize the mean individual income rate $G(\pi)$ for a small group of drivers:

$$G(\pi) = \frac{1}{k} \tilde{O}_{==1} \frac{\acute{I}_{=} J_{\cdot}^{(=)}(\pi)}{\acute{I}_{=} L_{\cdot}^{(=)}(\pi)}$$

Table 2: Statistics of operation time and online time

	Operation time	Online time
mean	58.926749	90.234227
std	43.856806	69.664996
min	0.133333	0.250000
25%	25.766667	37.400000
50%	47.500000	71.600000
75%	80.983333	124.600000
max	382.800000	652.850000

where π is the repositioning policy, $J_{:}^{(=)}(\pi)$ is the total income for driver k on day n and $L_{:}^{(=)}(\pi)$ is the corresponding online time for the driver.

We propose a single-agent reinforcement learning to solve this problem which is slightly different from Lin et al. [5] Formally, we treat the reposition problem as an MDP problem G, defined by a tuple $G = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma$ are the state, action space, transition probability function, reward function, and a discount factor respectively. The definitions are given as follows:

- **Agent**: We maintain a global repositioning agent for all vehicles, which is easy to implement efficiently. A vehicle is eligible for repositioning if and only if meet specific requirements, e.g., being idle for over five minutes. Suppose each vehicle requests for repositioning every half an hour, and there will be 48 samples in a day. Therefore, we design a single agent queue for dispatching multiple vehicles, and the number of samples will be 480 when given 10 vehicles. Here, the single-agent works as a global dispatcher, which receives the repositioning requests from the vehicle. It should be noted that concurrent repositioning requests do not exist in the environment of this competition. However, in real applications, the single agent can return an action list, such as the actions that can be matched with concurrent repositioning requests.
- **State** $s_C \in \mathcal{S}$: The state defined in this paper contains two parts, the global state and the auxiliary information state. In this experiment, the whole study area is partitioned into a $N \times N$ square grid in reference to the scope the trip data covers, each element of which is called a region. We maintain a global state $\boldsymbol{g_t} \in \mathbb{R}^{3\times \# \times \#}$, where t is the current time. At each time t, the three channels of g_C records the number of available vehicles, orders, and the sum of orders' rewards. Note that since the dispatch window is only two seconds, to mitigate the sparsity of the data, we introduced a smoothing operation where the sum of the last 10 minutes of data is recorded in each channel. For each vehicle repositioning request i at time t, the coordinate of the current location $\boldsymbol{l}_C^{ß} \in \mathbb{R}^2$ of the dispatched vehicle is used as an auxiliary information state. In addition, the current time t (mapped to $[-1, 1]$ uniformly) is also used as an auxiliary information state. Finally, the state of each vehicle repositioning request i at time t, is defined as

$$s_C^{ß} = \{\boldsymbol{g_t}, \boldsymbol{l}_C^{ß}, t\}$$

where vehicle repositioning requests in the same time share the same global state $\boldsymbol{g}_C$.

- **Action** $a_C \in \mathcal{A}$: When determining the strategy for vehicle repositioning request i at time t, the action space $\mathcal{A}$ specifies where the vehicle will be repositioned to at the following time. Lin et al. [5] defined seven discrete actions allocating the agent to one of its six neighboring grid items or staying in the current grid item. In comparison, we do not use local action, which limits the vehicle in the neighborhood of its current location. Instead, a global pruned action space, which encompasses a set of M discrete actions denoted by $\{k\}_{:=1}^{"}$, is used. Here, M is the total number of hexagonal grids, where grid items with few orders are filtered out based on historical data. According to experiment results, this action space appears to be more efficient, helping vehicles to avoid traveling to distant outskirts where there are no or sparse orders. Details on the policy of action selection will be elucidated in Section 4.3.
- **Reward function** $\mathcal{R}$: The agent aims at maximizing its expected discounted return, formulated as:

$$\mathbb{E}\left[\sum_{:=0}^{\tilde{\Theta}} \gamma^{:} r_{C+:}^{ß}\right]^{\#}$$

where $r_C^{ß}$ denotes the reward obtained from performing action $a_C^{ß}$, which can computed as:

$$r_C^{ß} = \sum_{9=1}^{\tilde{\Theta}} (o_9 - pe_9)$$

where m denotes the number of received orders before the next repositioning request, and p is a constant factor determined by global statistic. o_9 and e_9 denotes the order revenue and picking up eta respectively.

4 IMPLEMENTATION

In this section, we will introduce the implementation details of the simulator, choice of action space, dispatching policy, and reposition agent. The primary procedures of our solution can be summarized as follows:

- Julia Simulator,
- Greedy dispatching policy sorts the reward of orders in a unit time, and a strategy for declining the order is prepared,
- The action space is not limited to the local region where vehicles currently locate,
- Double Deep Q-learning for training the agent.

Table 3: Hyperparameter for the final submission

Description	value
state region latitude	$\phi \in [30.5N, 30.88N]$
state region longitude	$\lambda \in [103.89E, 104.28E]$
number of vehicles	$N_E = 20000$
order sample factor	$p_1 = 5.0$
grid resolution	$N = 30$
move penalty	$p = 0.002$
number of action	$M = 140$

Hyperparameters used in the final solution are listed in Table 3.

4.1 Simulator

The construction of learning environment is one of the fundamental challenges faced by the application of RL algorithms in reality, and building tailored simulators for the environment is a common practice in traffic research. For better performance, we write the simulator using the Julia programming language. The simulator can be decomposed into three different components, i.e., `Vehicle`, `Data` and `Simulator`, which are all encapsulated as structs. *Tc fgajc* (see Figure 3) is the base data structure storing the reward, availability and other information of each vehicle. `Data` is used for raw file processing, while `Simulator` is for the running of the simulator.

```
mutable struct Vehicle
    grid_id::Union{String,Nothing}
    total_reward::Union{Float32,Nothing}
    available_time::Union{Int64,Nothing}
    idle_time::Union{Int64,Nothing}
    idle_grid_id::Union{String,Nothing}
    location::NamedTuple{(:latitude,:longitude),
                         Tuple{Float64,Float64}}
    offline_time::Union{Int64,Nothing}
    id::Int64
end
```

Figure 3: The code snap for the definition of vehicle.

All the environment dynamics are consistent with the description listed in the competition introduction. In this example, we split one day into $T = 144$ time intervals (one for 10 minutes). Orders generated at the current time interval are bootstrapped by a sampling factor p_1 from real orders that occurred in the same time interval. At each full hour, each vehicle will update its state stochastically, either offline (i.e., off from service) or online (i.e., start receiving order requests). The updated probability p is determined based on statistics of historical data. The number of vehicles N_E and p_1 are critical hyperparameters that affect the relationship between supply and demand. We use hourly gross merchandise volume (GMV, i.e., the value of all the orders served) curve from the online submission to calibrate N_E and p_1. The correlation of GMV between our simulator and online data is demonstrated in Figure 4.

The major advantages of our simulator are (1) computing effiency and (2) easiness for debugging. Thanks to Just-In-Time (JIT) compilation and parallel computing capability of the Julia programming language, we achieve around 10 times faster speed compared with our simulator implementation in Python. Visualization, as shown in Figure 5, is also critical for RL training and is helpful to understand and debug the agent. It helps a lot to justify less desired actions and gives us more hints about the choice of action spaces and how to train the agent.

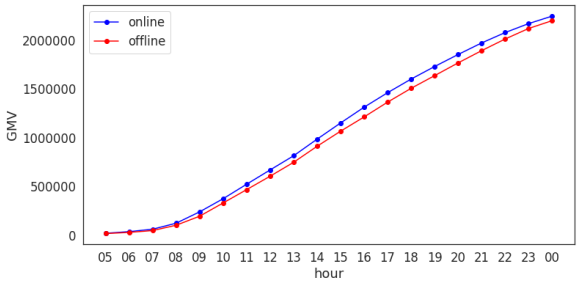


Figure 4: The simulator calibration in terms of GMV (R = 0.9956). The red curves plot the GMV values of online data while the blue are simulated result where p_1 = 5.0 and N_E = 20000.

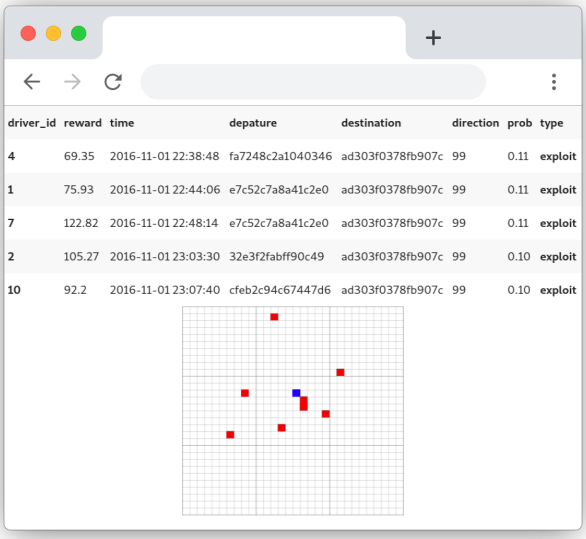


Figure 5: A simple training movement captured from our visualization tool. Grids in blue and red represent the reposition destinations and origins respectively. This visualization shows a over fitting or less desired scenario that all destination are the same.

4.2 Choice of dispatching policy

Although we are allowed to deploy both dispatching and repositioning agent, a greedy policy is adopted to dispatch orders and drivers for simplicity. In general, at each time window, we sort all order-driver pairs by reward per unit time in descending order, and the larger order-driver pair are more likely to be selected. Also, we take the order destination as a restricted factor. In other words, when the destination is undesirable, some orders will be declined even if there are available drivers for this order.

4.3 Choice of action space

The choice of action space $\mathcal{A}$ will highly influence not only the performance but also the stability of the result. Typical action spaces are listed below.

- **Full global action space**:
  vehicles can be repositioned to any grid without any restriction,
- **Pruned global action space**:
  vehicles can be repositioned to pre-defined grid set,
- **Local action space**:
  vehicles can be repositioned to grids that are closed to the current location by a pre-defined radius.

Based on our offline tests and online submissions, local action space performs worse among those three. This can be attributed to that our single-agent setting cannot catch up with the information on vehicle location very well. Full global action space is the desired action space, and however, is restrained by limited computational capability. Therefore, pruned global action space, covering a small subset (hot regions) of the full study area, is finally used.

4.4 Reposition agent

We apply Double deep Q-learning [16] for this task. Figure 6 illuminates our network architecture for policy network. To relief the gap between the unknown online environment and ours, we normalize the global state $\boldsymbol{s}_G$ by min-max scaler and map it to $[0, 1]$. We use simple ϵ-greedy to train the policy network and the training process is similar to that in pytorch DQN tutorial [13].

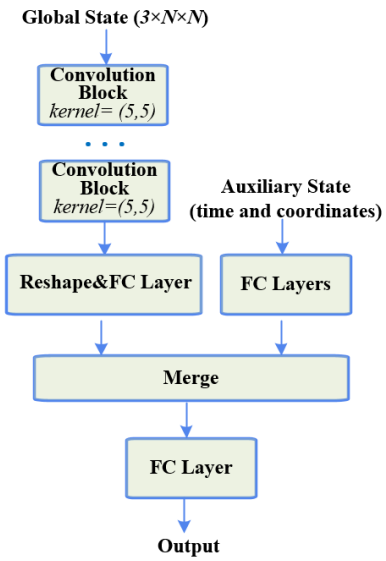


Figure 6: Policy Network Architecture

REFERENCES

[1] Long He, Zhenyu Hu, and Meilin Zhang. 2020. Robust Repositioning for Vehicle Sharing. *K_l sd_arspgle $ Qcptgac Mncp_rgmlq K_l_eckclr* 22, 2 (2020), 241–256. https://doi.org/10.1287/msom.2018.0734

[2] John Holler, Risto Vuorio, Zhiwei Qin, Xiaocheng Tang, Yan Jiao, Tiancheng Jin, Satinder Singh, Chenxi Wang, and Jieping Ye. 2019. Deep Reinforcement Learning for Multi-Driver Vehicle Dispatching and Repositioning Problem. In *0./7 GCCC Glrcpl_rgml_j Amldcpclac ml B_r_ Kglgle &&ABK'*. IEEE, Beijing, China, 1090–1095. https://doi.org/10.1109/ICDM.2019.00129

[3] Jiarui Jin, Ming Zhou, Weinan Zhang, Minne Li, Zilong Guo, Zhiwei Qin, Yan Jiao, Xiaocheng Tang, Chenxi Wang, Jun Wang, Guobin Wu, and Jieping Ye. 2019. CoRide: Joint Order Dispatching and Fleet Management for Multi-Scale Ride-Hailing Platforms. In *Npmaccbgleq md rfc 06rf ?AK Glrcpl_rgml_j Amldcpclac ml Gldmpk_rgml_lb Illmujcbec K_l_eckclr*. ACM, Beijing China, 1983–1992. https://doi.org/10.1145/3357384.3357978

[4] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. 2018. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In *Npmaccb+gleq md rfc 02rf ?AK QGEIBB Glrcpl_rgml_j Amldcpclac ml Illmujcbec Bgqamtcpw $ B_r_ Kglgle*. ACM, London United Kingdom, 1774–1783. https://doi.org/10.1145/3219819.3219993

[5] Lei Lin, Zhengbing He, and Srinivas Peeta. 2018. Predicting Station-Level Hourly Demand in a Large-Scale Bike-Sharing Network: A Graph Convolutional Neural Network Approach. *Rp_lqnmpr_rgml Pcqc_paf N_pr A8 Ckcpegle Rcaflmjmegcq* 97 (2018), 258–276. https://doi.org/10.1016/j.trc.2018.10.011

[6] Yang Liu, Zhiyuan Liu, Cheng Lyu, and Jieping Ye. 2019. Attention-Based Deep Ensemble Net for Large-Scale Online Taxi-Hailing Demand Prediction. *GCCC Rp_lq_argmlq ml Glrcjjgeclr Rp_lqnmpr_rgml Qwqrckq* (2019), 1–10. https://doi.org/10.1109/TITS.2019.2947145

[7] Yang Liu, Cheng Lyu, Zhiyuan Liu, and Dacheng Tao. 2019. Building Effective Short Video Recommendation. In *0./7 GCCC Glrcpl_rgml_j Amldcpclac ml Ksjrg+kcbg_ $ Cvnm Umpiqfmnq &&AKCU'*. IEEE, Shanghai, China, 651–656. https://doi.org/10.1109/ICMEW.2019.00126

[8] Yang Liu, Fanyou Wu, Baosheng Yu, Zhiyuan Liu, and Jieping Ye. 2019. Building Effective Large-Scale Traffic State Prediction System: Traffic4cast Challenge Solution. *_pVgt8/7//,.3477 Yaqf* (2019). arXiv:1911.05699 [cs]

[9] Zhiyuan Liu, Yang Liu, Cheng Lyu, and Jieping Ye. 2020. Building Personalized Transportation Model for Online Taxi-Hailing Demand Prediction. *GCCC Rp_lq+_argmlq ml Aw`cplcrgaq* In press (2020).

[10] Tai-Yu Ma, Saeid Rasulkhani, Joseph Y.J. Chow, and Sylvain Klein. 2019. A Dynamic Ridesharing Dispatch and Idle Vehicle Repositioning Strategy with Integrated Transit Transfers. *Rp_lqnmpr_rgml Pcqc_paf N_pr C8 Jmegqrgaq _lb Rp_lq+nmpr_rgml Pctgcu* 128 (2019), 417–442. https://doi.org/10.1016/j.tre.2019.07.002

[11] Chao Mao, Yulin Liu, and Zuo-Jun (Max) Shen. 2020. Dispatch of Autonomous Vehicles for Taxi Services: A Deep Reinforcement Learning Approach. *Rp_lq+nmpr_rgml Pcqc_paf N_pr A8 Ckcpegle Rcaflmjmegcq* 115 (2020), 102626. https://doi.org/10.1016/j.trc.2020.102626

[12] Rahul Nair and Elise Miller-Hooks. 2011. Fleet Management for Vehicle Sharing Operations. *Rp_lqnmpr_rgml Qagclac* 45, 4 (2011), 524–540. https://doi.org/10.1287/trsc.1100.0347

[13] Adam Paszke. [n.d.]. REINFORCEMENT LEARNING (DQN) TUTORIA. https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html. Accessed: 2020-05-01.

[14] Debjit Roy, Jennifer A. Pazour, and René de Koster. 2014. A Novel Approach for Designing Rental Vehicle Repositioning Strategies. *GGC Rp_lq_argmlq* 46, 9 (2014), 948–967. https://doi.org/10.1080/0740817X.2013.876129

[15] Dong-Ping Song and Christopher F. Earl. 2008. Optimal Empty Vehicle Repositioning and Fleet-Sizing for Two-Depot Service Systems. *Cspmnc_l Hmspl_j md Mncp_rgml_j Pcqc_paf* 185, 2 (2008), 760–777. https://doi.org/10.1016/j.ejor.2006.12.034

[16] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *⇔gprgcrf ???G amldcpclac ml _prg`ag_j glrcj+jgeclac*.

[17] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In *⇔c ⇔gprw+Qcamlb ???G Amldcpclac ml ?p+rg`ag_j Glrcjjgeclac &???G+/6'*. AAAI, New Orleans, Louisiana, USA, 2589–2595. arXiv:1802.08714

[18] Ming Zhou, Jiarui Jin, Weinan Zhang, Zhiwei Qin, Yan Jiao, Chenxi Wang, Guobin Wu, Yong Yu, and Jieping Ye. 2019. Multi-Agent Reinforcement Learning for Order-Dispatching via Order-Vehicle Distribution Matching. In *Npmaccbgleq md rfc 06rf ?AK Glrcpl_rgml_j Amldcpclac ml Gldmpk_rgml_lb Illmujcbec K_l_ec+kclr*. ACM, Beijing China, 2645–2653. https://doi.org/10.1145/3357384.3357799