Knowledge-Based Systems 205 (2020) 106302

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Spatio-temporal feature fusion for dynamic taxi route recommendation via deep reinforcement learning

Shenggong Ji [a], Zhaoyuan Wang [a], Tianrui Li [a,b,*], Yu Zheng [a,c,d,*]

[a] School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China
[b] National Engineering Laboratory of Integrated Transportation Big Data Application Technology, Southwest Jiaotong University, Chengdu 611756, China
[c] JD Intelligent Cities Research, JD.com, Beijing 100176, China
[d] JD Intelligent Cities Business Unit, JD Digits, Beijing 100176, China

ARTICLE INFO

Article history:
Received 29 February 2020
Received in revised form 6 July 2020
Accepted 20 July 2020
Available online 25 July 2020

Keywords:
Spatio-temporal feature fusion
Sequential decision making
Taxi route recommendation
Deep reinforcement learning
Transportation

ABSTRACT

Dynamic taxi route recommendation aims at recommending cruising routes to vacant taxis such that they can quickly find and pick up new passengers. Given citizens' giant but unbalancing riding demand and the very limited taxis in a city, dynamic taxi route recommendation is essential for its ability to alleviate the waiting time of passengers and increase the earning of taxi drivers. Thus, in this paper we study the dynamic taxi route recommendation problem as a sequential decision-making problem and we design an effective two-step method to tackle it. *First*, we propose to consider and extract multiple real-time spatio-temporal features, which are related with the easiness degree of vacant taxis picking up new passengers. *Second*, we design an adaptive deep reinforcement learning method, which learns a carefully designed deep policy network to better fuse the extracted spatio-temporal features such that effective route recommendation can be done. Extensive experiments using real-world data from San Francisco and New York are conducted. Comparing with the state-of-the-arts, our method can increase at least 15.8% of average earning for taxi drivers and reduce at least 29.6% of average waiting time for passengers.

1. Introduction

Taxi is an important commuting mode in our daily lives. Every day, in cities like New York and Beijing, there are hundreds of thousands of people waiting to take taxis [1,2]. In contrast to citizens' giant riding demand, the number of taxis in a city, however, is very limited. Even worse, after dropping off passengers, taxis need to spend a lot of time on cruising vacantly to find new passengers, since passengers' riding demand is spatially and temporally unbalancing [3,4]. Consequently, the difficulty of taking taxis has become a common phenomenon in big cities [3,5,6]. Improving the transportation efficiency of taxis has become an urgent issue. Dynamic taxi route recommendation [3,7], i.e., recommending cruising routes to vacant taxis, provides an feasible way. Specifically, as shown in Fig. 1(a), dynamic taxi route recommendation is to recommend a route (defined as a sequence of road segments, e.g. routes r_1, r_2, r_3, r_4) to the vacant taxi 1 such that by following the recommended route, taxi 1 can quickly find a new passenger. It is a sequential decision-making problem [8] and perhaps more than one round of route recommendation is necessary before a vacant taxi picks up new passengers. Taxi route recommendation has many benefits, e.g. bringing taxi drivers more revenues, reducing waiting time for passengers.

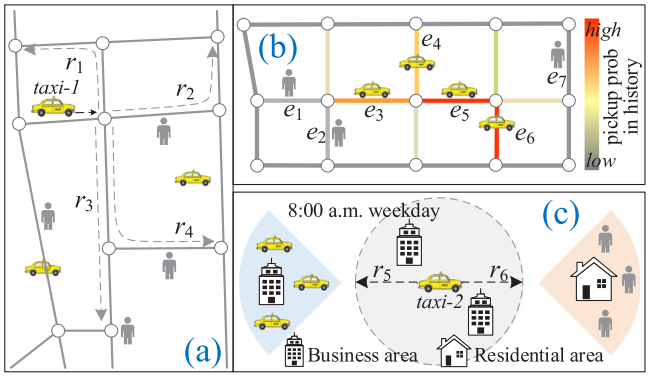


Fig. 1. Dynamic taxi route recommendation.

* Corresponding authors.
E-mail addresses: shenggongji@163.com (S. Ji), wang_zhaoyuan@foxmail.com (Z. Wang), trli@swjtu.edu.cn (T. Li), msyuzheng@outlook.com (Y. Zheng).

https://doi.org/10.1016/j.knosys.2020.106302
0950-7051/

However, dynamic taxi route recommendation is challenging, given the following two reasons. *First*, taxi route recommendation is related with multiple real-time spatio-temporal features, which are non-trivial to formulate. Although there have already been several literature studying dynamic taxi route recommendation, real-time spatio-temporal features have neither been comprehensively considered nor formulated [3,7,9]. In this work, we consider and formulate the following internal and external features as our real-time spatio-temporal features.

- *Real-time*. The reason that we should consider real-time features is that the dynamic recommendation will significantly affect the status of the system. For the example in Fig. 1(b), road segments e_3, e_4, e_5, e_6 (in red) have high pickup probabilities in history. However, after some vacant taxis have been recommended to them, most passengers have been picked up and then these routes' real-time pickup probabilities become small. At this moment, if we still conduct recommendation based on historical (static) pickup probability, we will continuously recommend vacant taxis to these road segments, making them cannot find any passenger there. Therefore, we should consider the real-time features.
- *Internal*. For each possible route for recommendation, the real-time internal features are features related with the easiness degree of vacant taxis picking up passengers on the route, e.g. the number of vacant taxis currently on the route, the (estimated) number of current waiting passengers on the route. Note that prior literature considers historical/static internal features [3,9], not real-time internal features.
- *External*. The real-time external features of a route are features related with the easiness degree of vacant taxis picking up passengers in the future (not on this route). For example, as shown in Fig. 1(c), the vacant taxi 2 is currently in a business area at the rush hour of a weekday morning (e.g. 8:00 a.m.), during which most people go from residential areas to business areas. The gray circle in Fig. 1(c) indicates the maximum length of a candidate route, e.g. five road segments [7]. In this case, all candidate routes (e.g. r_5, r_6) still locate in the business area without any waiting passengers. That is, all candidate routes have highly similar real-time internal features, e.g. their real-time pickup probabilities are all close to zero. As a result, if considering only the real-time internal features on each route, recommending r_5 or r_6 makes no difference. However, as Fig. 1(c) shows, r_6 is clearly better than r_5 since r_6 leads to a residential area with many waiting passengers, while r_5 leads to another business area with no waiting passenger. That is, for a route, besides its real-time internal features, the real-time external features are vital, too. To our best knowledge, the real-time external features have not yet been considered by the state-of-the-arts [3,7,9].

Second, to fuse the above spatio-temporal features so that better taxi recommendation can be done is challenging. In particular, we do not have label data. Given the features of all candidate routes, there is no labeled optimal recommendation decision. As a result, supervised learning-based feature fusion methods (a mapping from features of each route to the corresponding recommended route) cannot be used in this problem.

To address the above issues, in this paper, we design a deep reinforcement learning method to fuse the real-time internal and external spatio-temporal features into the dynamic route recommendation of vacant taxis. Overall, our contributions are three-fold.

- To our best knowledge, we are the first to consider both the real-time internal and external spatio-temporal features for dynamic taxi route recommendation. In addition, we provide formal definitions and formulations for these features.
- We design a deep reinforcement learning method to fuse the extracted features to do the dynamic route recommendation for vacant taxis. Specifically, a deep neural network (deep policy network) is carefully designed to fuse the extracted features. Then we develop an adaptive deep reinforcement learning method to learn the deep policy network to do better recommendation.
- Extensive experiments are conducted using data collected from San Francisco and New York. Experiment results show that comparing with existing methods, our method is able to improve at least 15.8% of earning for taxi drivers and reduce at least 29.6% of waiting time for passengers.

The rest of this paper is organized as follows. In Section 2, we introduce the related work. Section 3 provides an overview of our method for dynamic taxi route recommendation. Section 4 formally defines and formulates our real-time internal and external spatio-temporal features. In Section 5, we design a deep reinforcement learning method to learn a carefully designed deep policy network to fuse the extracted features and to do the recommendation. Experiments are conducted in Section 6, followed by the conclusion of this paper in Section 7.

2. Related work

2.1. Taxi route recommendation

Prior taxi route recommendation methods can be categorized into two groups, according to different problem definitions. The first group of methods [3,7,9,10] aim at recommending a detailed route (i.e., a sequence of road segments) to each vacant taxi. The method proposed in this work belongs to this group. The second group of methods [4,5,11] recommend an area to a vacant taxi, e.g. a grid [4], a zone [11], or a road cluster [5]. That is, the second group of methods do not recommend a detailed route to a taxi. In the future, we plan to study a variant of our method such that it can be applied to recommend an area to a taxi, too.

More specifically, Yuan et al. [3] proposed a probabilistic model to do the taxi route recommendation. They extracted the probability of vacant taxis finding new passengers on each route from the data in history. Then they recommended a route with the maximal pickup probability to a vacant taxi. Besides, they also recommended locations to passengers such that passengers can be picked up quickly. Qu et al. [7] firstly calculated the net profit of each candidate route using the data in history, and then provided an effective algorithm to recommend routes with more net profit. Garg et al. [9] developed a Monte Carlo Tree Search method aiming at minimizing the vacant driving distance for drivers. Similarly, Luo et al. [10] also proposed a taxi route recommendation method to minimize the average vacant driving distance. Rong et al. [4] proposed a Markov decision process method to recommend a grid for each vacant taxi. Based on the pickup probability and rewards obtained in history, they learned the state value and state–action value for each state and each state–action pair. Then they conducted the recommendation using the learned state–action value. Verma et al. [11] developed a table Q-learning method to recommend a zone to each vacant taxi. To learn the state–action q-value, they leveraged Monte Carlo sampling method. Likewise, they conducted recommendation based on the learned q-value.

Despite of the different problem definitions and the different methods used, prior taxi route recommendation methods have a common drawback. That is, when conducting recommendation,

they do not consider real-time features in the taxi system. For example, [3] recommends a route based on the pickup probability of each route in history. Likewise, [7] recommends a route based on the historical net profit of a route. Others [5,9,11] do not directly consider any features at all. However, the real-time features will significantly affect the recommendation performance. Therefore, in this work we propose to consider the real-time features (internal features and external features). To the best of our knowledge, external features have not been considered. We propose a deep policy network learned by deep reinforcement learning to better fuse these features.

2.2. Taxi dispatching and taxi sharing

Besides taxi route recommendation, taxi dispatching [12–14] and taxi sharing [15–17] can also improve taxis' transportation efficiency. Taxi dispatching is to select a *vacant* taxi to pick up a passenger who has sent a request for riding. For example, Zhang et al. [12] developed a combinatorial optimization model to do the taxi dispatching such that the average driver acceptance rate for all orders can be maximized. Xu et al. [13] proposed to learn the state value for each hexagon grid and then do the optimization considering the learned state value. Taxi sharing is to select an *occupied* taxi to pick up a passenger who has sent a riding request. In taxi sharing, different passengers can share one taxi. Ma et al. [15] proposed a T-share system, which is able to do large-scale taxi sharing based on a carefully designed spatio-temporal index. Santos et al. [16] modeled the taxi-sharing problem as an optimization problem and provided heuristic methods to maximize the number of shared trips. Taxi dispatching and taxi sharing usually happen after a passenger sends a riding request to online taxi platforms [12,17], e.g. Uber, Lyft, and Didi. For online taxi platforms, taxi route recommendation happens when there is no passenger sending riding requests and vacant taxis need to cruise in empty before new passengers come, especially at off-peak time. For traditional taxi systems, passengers do not send their requests online but call taxis by the roadside, thus taxi route recommendation is always necessary [3,7]. In fact, taxi route recommendation, taxi dispatching, and taxi sharing have different application scenarios and thus can complement with each other to better enhance taxis' transportation efficiency.

2.3. Transportation

The research on the taxi route recommendation belongs to the transportation research in urban area. In addition, there exist many other transportation-related problems. For example, Lee et al. [18] proposed to mine the traffic bottlenecks in the road networks of a city using a three-phase spatio-temporal traffic bottleneck mining model. The mining of the traffic bottlenecks can help alleviate the traffic congestion in road networks. Pan et al. [19] modeled the human behavior as a sequential decision-making process, and proposed to study the dynamic human preference using inverse reinforcement learning. Using the method, deeper analysis results for the behavior of taxi drivers have been obtained. Yu et al. [20] proposed an effective deep learning framework for traffic forecasting, which is able to accurately predict the mid-and-long term traffic flow in a city. The proposed framework can well fuse the spatial and temporal correlations between data in different domains.

2.4. Deep reinforcement learning

Recent advances in deep learning [21,22] have significantly spurred the success of deep reinforcement learning [8,23]. Many effective deep reinforcement learning methods have been developed, such as deep Q-learning [24], deep double Q-learning [25], dueling [26], proximal policy optimization [27], and so on. Deep reinforcement learning has achieved excellent performance in many decision making problems, for example, Go [28,29], robotics [30,31], Atari 2600 games [24], early classification [32], ambulance redeployment [33], energy saving [34], etc. In this work, we design an adaptive and novel deep reinforcement learning method to do the dynamic taxi route recommendation.

3. Overview

3.1. Preliminary

Definition 1 (*Road Network*)**.** A road network is a directed road network, denoted by $G = \langle V, E\rangle$, where V denotes all road intersections (vertices) and E refers to all road segments (edges). For each directed edge $e \in E$, it contains two vertices, i.e., $e = \langle v_{\text{in}}, v_{\text{out}}\rangle$, which means there is a directed edge from vertex v_{in} to vertex v_{out}. Based on our definition, $\langle v_{\text{in}}, v_{\text{out}}\rangle$ and $\langle v_{\text{out}}, v_{\text{in}}\rangle$ are two different directed edges. Same with literature [7,9], the dynamic taxi route recommendation problem in this work is also defined on the road network G of a city.

Definition 2 (*Request*)**.** A recommendation request *req* from a taxi is a tuple, denoted by $req = \langle e, t\rangle$. e denotes the current located edge of the taxi. $e.v_{\text{in}}$ is the vertex through which the taxi enters the edge e and $e.v_{\text{out}}$ is the vertex through which the taxi will leave the edge e. t refers to the time stamp at which the request is sent to our backend. In this paper, for each vacant taxi, it will continuously send its request until it picks up new passengers. Usually, a vacant taxi has to send a few requests (i.e., a few rounds of recommendation) before it picks up new passengers.

Definition 3 (*Route*)**.** A recommendation route for a request $req = \langle e, t\rangle$ is a sequence of road segments (edges), denoted by $r = e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_l$ (see Fig. 1(a)). That is, after a recommendation request *req* from a taxi is sent to our taxi route recommendation backend, we will return a recommendation route to the taxi. l is the recommendation route length, i.e., the number of road edges in r, e.g. $l = 4$. The edges $e_1, e_2, \ldots, e_l$ in r should meet the following constraints:

$$\begin{cases} e_1.v_{\text{in}} = req.e.v_{\text{out}} \\ e_2.v_{\text{in}} = e_1.v_{\text{out}} \\ e_3.v_{\text{in}} = e_2.v_{\text{out}} \\ \cdots \\ e_l.v_{\text{in}} = e_{l-1}.v_{\text{out}} \end{cases} \tag{1}$$

For a request *req*, we denote the entire candidate recommendation routes by $\mathcal{R}(req) = \{r_1, r_2, \ldots, r_Q\}$. Q refers to the cardinality of $\mathcal{R}(req)$, and for different *req*, Q can be different. Each route $r \in \mathcal{R}(req)$ satisfies the constraints (1). Besides, we denote $r_*(req)$ as the optimal recommendation route for request *req*.

3.2. Problem definition

Dynamic taxi route recommendation problem: the dynamic taxi route recommendation problem aims at finding the optimal route $r_*(req)$ from the candidate routes $\mathcal{R}(req)$ for each coming recommendation request *req*, such that by following the recommended routes, taxis' average vacant cruising time can be

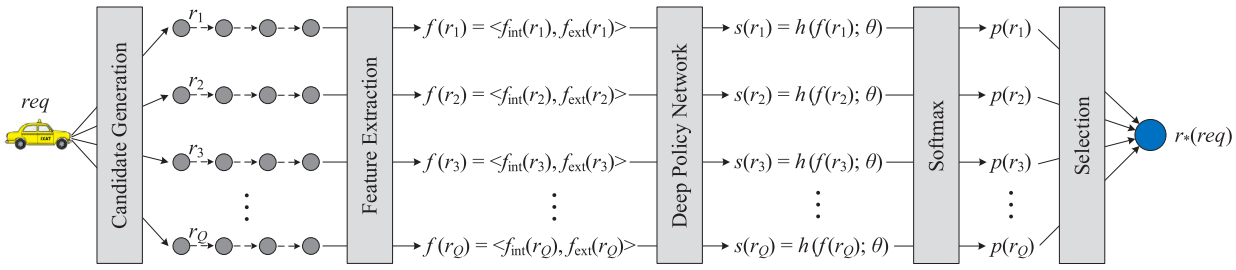


Fig. 2. The general framework of our dynamic taxi route recommendation.

minimized. That is, we are going to learn an effective taxi route recommendation policy.

Besides minimizing taxis' vacant cruising time, there exist other objective functions, e.g. maximizing taxis' pickup probability [3], maximizing taxi drivers' profit [7], minimizing taxis' vacant cruising distance [9], etc. Although in the problem definition, our objective is to minimize the average vacant cruising time of taxis, the proposed recommendation method can easily be adapted to other objective functions. In our experiments, we set other objective functions as the metrics to measure our taxi route recommendation method.

3.3. Framework

As presented in Fig. 2, when a recommendation request $req = \langle e, t \rangle$ comes to our backend, we have three main components to do the recommendation, i.e., Candidate Generation, Feature Extraction, and Deep Policy Network. Below, we detail each component.

Candidate Generation is to generate all candidate recommendation routes $\mathcal{R}(req) = \{r_1, r_2, \ldots, r_Q\}$ for the request req, as shown in Fig. 2. To generate all candidate routes is easy, and we can directly use some search algorithms in graph [35], e.g. the popularly used breadth-first and depth-first search methods. Note that, for different req, Q can be different.

Feature Extraction is to extract recommendation-related features $f(r_q)$ for each candidate recommendation route $r_q \in \mathcal{R}(req)$. As discussed before, we consider the real-time internal and external spatio-temporal features of a route r_q, i.e.,

$$f(r_q) = \left\langle f_{\text{int}}(r_q), f_{\text{ext}}(r_q) \right\rangle, \tag{2}$$

where $f_{\text{int}}(r_q)$ denotes the internal features of route r_q and $f_{\text{ext}}(r_q)$ refers to r_q's external features. We formally define and formulate the real-time internal and external spatio-temporal features in Section 4.

Deep Policy Network (learned with Deep Reinforcement Learning) maps each candidate route r_q's features $f(r_q)$ to the score $s(r_q)$ of each route r_q, that is,

$$s(r_q) = h\left(f(r_q); \theta\right), \tag{3}$$

where h denotes the nonlinear mapping of the deep policy network from a route r_q's features $f(r_q)$ to the score $s(r_q)$ of the route r_q. The θ refers to the parameters in the deep policy network. The network structure is detailed in Fig. 6 in Section 5. That is, we provide a deep policy network with a carefully designed structure to fuse the real-time internal and external features. Next, based on the obtained score $s(r_q)$ of each route r_q, we can calculate the probability $p(r_q)$ of recommending each route r_q using the softmax function:

$$p(r_q) = \frac{\exp\left(s(r_q)\right)}{\sum_{q'=1}^{Q} \exp\left(s(r_{q'})\right)}. \tag{4}$$

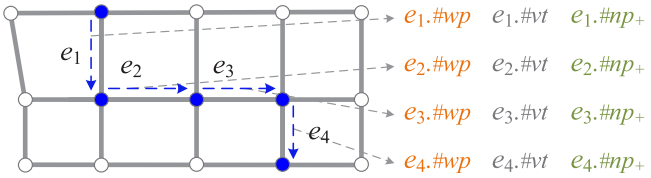


Fig. 3. Internal features $f_{\text{int}}(r)$ of $r = e_1 \to e_2 \to e_3 \to e_4$.

Finally, we recommend route $r_*(req)$ to the vacant taxi according to each route r_q's recommendation probability $p(r_q)$. The larger the score $s(r_q)$, the higher probability r_q is recommended. To learn an optimal deep policy network, we will design a deep reinforcement learning method in Section 5.

4. Feature extraction

4.1. Real-time internal spatio-temporal features

For each candidate route $r = e_1 \to e_2 \to \cdots \to e_l \in \mathcal{R}(req)$ of a request req, we expect the real-time internal features $f_{\text{int}}(r)$ can reflect the real-time pickup probability on r. Clearly, the pickup probability on r is related with the pickup probability on r's edges $e_1, e_2, \ldots, e_l$. Thus, we first consider each edge e_i's real-time pickup probability, and we propose to consider the following pickup probability-related features for each edge e_i:

1. Estimated number of current waiting passengers at this edge, $e_i.\#wp$.
2. Number of current vacant taxis at this edge, $e_i.\#vt$.
3. Estimated number of net coming passengers at this edge in the next time period (e.g. next half an hour), $e_i.\#np_+$.

These features will be detailed later. Obviously, more waiting passengers, less vacant taxis, and more net coming passengers will lead to higher pickup probability on edge e_i.

Then, we can define a route r's real-time internal features $f_{\text{int}}(r)$ as the concatenation of features on $e_1, e_2, \ldots, e_l \in r$, i.e.,

$$f_{\text{int}}(r) = \begin{bmatrix} e_1.\#wp & e_1.\#vt & e_1.\#np_+ \\ e_2.\#wp & e_2.\#vt & e_2.\#np_+ \\ \cdots & \cdots & \cdots \\ e_l.\#wp & e_l.\#vt & e_l.\#np_+ \end{bmatrix}. \tag{5}$$

Fig. 3 is an example of the internal features of a route $r = e_1 \to e_2 \to e_3 \to e_4$, where $l = 4$. Apparently, for a vacant taxi, we expect to recommend it a route r with more waiting passengers (i.e., larger $e_1.\#wp, \ldots, e_l.\#wp$), less vacant taxis (i.e., smaller $e_1.\#vt, \ldots, e_l.\#vt$), and more coming net passengers (i.e., larger $e_1.\#np_+, \ldots, e_l.\#np_+$). Below, we formulate these features in detail.

4.1.1. $e_i.\#wp$

The number of current waiting passengers on edge e_i is unknown and should be estimated. We propose to utilize two values to represent $e_i.\#wp$, one as the estimated number of arrived passengers in the last time period (e.g. the last half an hour), denoted by $e_i.\#ap_-$, the other as the number of passengers picked up in the last time period, denoted by $e_i.\#pp_-$. $e_i.\#pp_-$ is a knowable value. Then, the number of current waiting passengers is the number of arrived passengers minus the number of passengers picked up in the last time period. That is, $e_i.\#wp$ is

$$e_i.\#wp = e_i.\#ap_- - e_i.\#pp_-. \tag{6}$$

To estimate the number $e_i.\#ap_-$ of arrived passengers at an edge e_i, many existing time series estimation methods can be used, e.g. autoregressive integrated moving average (ARIMA) models [36], linear regression models [37], deep neural networks [38–40], etc. Since the focus of this work is not on the estimation method, we learn an ARIMA model to do the estimation, which is effective, easy to implement, and widely used in prior literature [36].

4.1.2. $e_i.\#vt$

The number of vacant taxis at edge e_i is a knowable value since all taxis are equipped with wireless sensors, probing their real-time GPS locations and status (occupied or vacant). Thus, we can know the real-time number of vacant taxis at each edge. By considering this feature, all vacant taxis can be cooperated to do better recommendation. For example, if the edge e_i has enough vacant taxis, we will not recommend a route covering this edge to new vacant taxis. We can also avoid recommending the same route to many vacant taxis.

4.1.3. $e_i.\#np_+$

The estimated number of net coming passengers in next time period at edge e_i is

$$e_i.\#np_+ = e_i.\#ap_+ - e_i.\#dt_+, \tag{7}$$

where $e_i.\#ap_+$ is the estimated number of arriving passengers at edge e_i in next time period, and $e_i.\#dt_+$ refers to the estimated number of taxis that will drop passengers at edge e_i in next time period. Thus $e_i.\#np_+$ can reflect the future riding demand of passengers at edge e_i, i.e., the future pickup probability of edge e_i. The bigger the $e_i.\#np_+$, the higher probability of a taxi picking up a passenger in e_i. We also learn an ARIMA model to estimate $e_i.\#ap_+$ and $e_i.\#dt_+$.

4.2. Real-time external spatio-temporal features

As discussed before, the external features $f_{\text{ext}}(r)$ of a route r are to reflect the easiness degree of a vacant taxi picking up passengers in the future after it reaches the end of the recommended route r. Thus, we propose to consider the features nearby the end of route $r = e_1 \to \cdots \to e_l$, i.e., nearby the vertex $e_l.v_{\text{out}}$. Formally, we consider the features in grids surrounding $e_l.v_{\text{out}}$. A city can be segmented into many grids, each one of which has the same size (e.g. 300 m×300 m). For example, as shown in Fig. 4, $e_l.v_{\text{out}}$ is in grid g_{ij}, and we consider grid g_{ij}'s k orders of surrounding grids, $k = 0, 1, 2$, respectively. We denote $\mathcal{G}(r)$ as the surrounding grids of a route r's end.

Same with the real-time features considered in each edge, for each grid $g_{i'j'} \in \mathcal{G}(r)$, we consider the following features:

1. Estimated number of current waiting passengers in this grid, $g_{i'j'}.\#wp$.
2. Number of current vacant taxis in this grid, $g_{i'j'}.\#vt$.

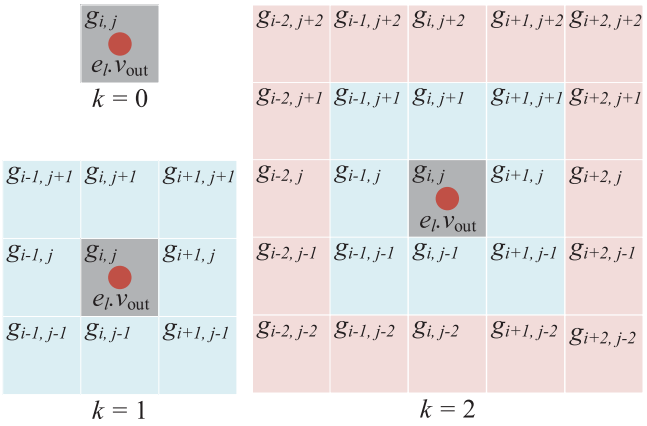


Fig. 4. Surrounding grids $\mathcal{G}(r)$ of route r's end, $k = 0, 1, 2$.

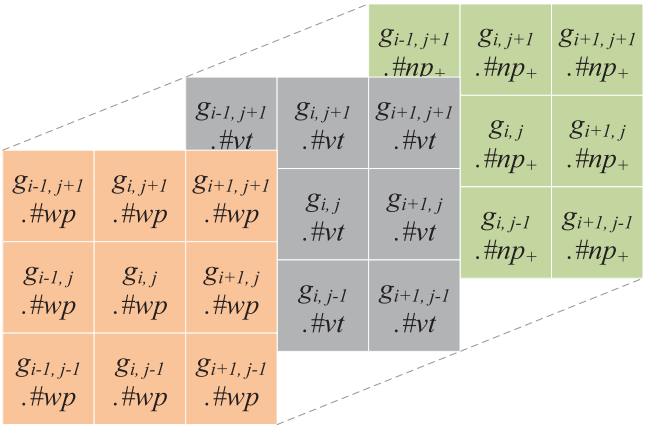


Fig. 5. External features $f_{\text{ext}}(r)$, $k = 1$.

3. Estimated number of net coming passengers in next time period in this grid, $g_{i'j'}.\#np_+$.

Clearly, these features also reflect the grid $g_{i'j'}$'s real-time and future pickup probability. Thus, if the surrounding grids of a route r's end have higher pickup probabilities, after a taxi reaches route r's end, it can easily find passengers in the future.

Formally, the real-time external features $f_{\text{ext}}(r)$ of a route r are defined as the concatenation of features in each grid $g_{i'j'} \in \mathcal{G}(r)$. Fig. 5 shows a route r's external features $f_{\text{ext}}(r)$ considering $k = 1$ order of surrounding grids being considered.

5. Deep reinforcement learning

In this section, we first introduce the structure of the proposed deep policy network. Then, we detail how to use deep reinforcement learning to learn the deep policy network to better fuse the extracted features and to do better recommendation.

5.1. Deep policy network

To better fuse the proposed real-time internal and external spatio-temporal features, we propose a deep neural network as the policy network, as demonstrated in Fig. 6. The input of the deep policy network includes the internal and external features of a route r, i.e., the upper left part of Fig. 6. The output is the score of the route, i.e., the $s(r)$ at the bottom of Fig. 6. The structure of the policy network is comprised of the following components.

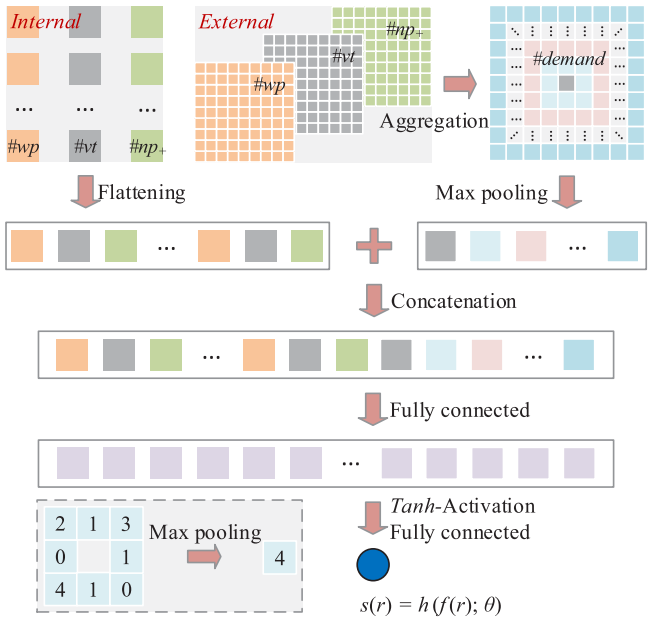


Fig. 6. The deep policy network: $f(r) \mapsto s(r) = h(f(r); \theta)$.

Flattening. Firstly, we transform the internal features $f_{\text{int}}(r)$ from a matrix Eq. (5) to a flattened layer.

Aggregation. For each grid $g_{i'j'} \in \mathcal{G}(r)$, it has three features $g_{i'j'}.\#wp$, $g_{i'j'}.\#vt$, and $g_{i'j'}.\#np_+$. Since the three features reflect the real-time and future pickup probability (i.e., riding demand) in grid $g_{i'j'}$, we can aggregate them as one:

$$g_{i'j'}.\#demand = (g_{i'j'}.\#wp - g_{i'j'}.\#vt) + \gamma \times g_{i'j'}.\#np_+. \tag{8}$$

$(g_{i'j'}.\#wp - g_{i'j'}.\#vt)$ indicates grid $g_{i'j'}$'s current riding demand, while $g_{i'j'}.\#np_+$ indicates the future demand. γ is a discount rate of future demand to current demand, and usually $\gamma \in [0, 1]$. After aggregation, for each grid $g_{i'j'} \in \mathcal{G}(r)$, as shown at the top right of Fig. 6, it contains only one feature $g_{i'j'}.\#demand$, indicating passengers' riding demand (pickup probability) in $g_{i'j'}$.

Max pooling. For each order of surrounding grids, we conduct max pooling operation. The bottom left corner of Fig. 6 shows the max pooling operation on the riding demand of the 1st-order grids. Max pooling can reduce the number of parameters needed to learn and in the meantime can avoid over-fitting [21]. For surrounding grids of the same order, the maximum demand is kept. After max pooling, we have k values, indicating the maximum riding demand of surrounding grids in each order (k orders).

Concatenation. The flattened internal features and k orders of surrounding grids' maximum riding demand are concatenated.

Fully connected. Two fully connected layers are then added to map the concatenated features into the score $s(r)$ of route r. After the first fully connected layer, a *Tanh* activation function is applied to conduct nonlinear mapping [21].

5.2. Learning deep policy network

To learn the parameters θ in the deep policy network (Fig. 6), we first model the dynamic taxi route recommendation as a reinforcement learning task [8,41]. Then, we learn the deep policy network using policy gradient [42].

5.2.1. Reinforcement learning task

Reinforcement learning deals with sequential decision-making problems in decision-making systems [8,41]. The basic framework of reinforcement learning can be modeled as Fig. 7, in

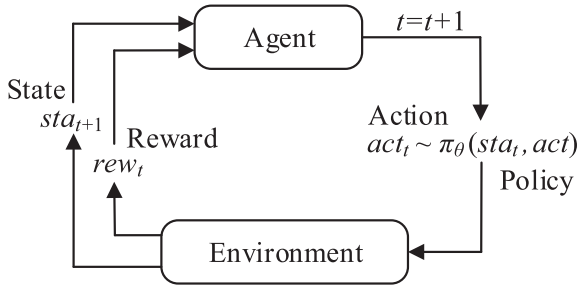


Fig. 7. Reinforcement learning framework.

which an agent interacts with the environment. More specifically, based on the current state sta_t of the environment, the agent makes an action act_t according to the policy π_θ. The θ in π_θ is the parameters, mapping the state–action pair (sta_t, act) into a probability of taking action act. In this work, the θ in π_θ is the θ in our deep policy network (see Fig. 6). Then, the agent obtains an instant reward rew_t and the state of the environment evolves, becoming sta_{t+1}. The goal of reinforcement learning is to learn an optimal policy π_θ, i.e., an optimal policy network in this work, such that by following the policy π_θ, the agent can obtain more rewards.

The dynamic taxi route recommendation problem can be modeled as a reinforcement learning task. The agent in reinforcement learning is a taxi in our taxi route recommendation problem while the environment is a taxi system in a city. Each taxi in the taxi system shares the same policy π_θ, i.e., the same policy network in Fig. 6.

State. When a taxi sends a request req for recommendation, we need to obtain the current state sta_t in the environment. The state sta_t includes all information in the current environment, e.g. the current recommendation request req, the current status of each taxi, the current status of passengers, and so on. Thus, sta_t can be denoted by

$$sta_t = (req, STATUS_{TAXIS}, STATUS_{PASSENGERS}, \ldots). \tag{9}$$

Action. The action act_t is a potential recommendation route r_q for the current request req, i.e.,

$$act_t \in \mathcal{R}(req), \tag{10}$$

where $\mathcal{R}(req)$ is the set of all possible candidate recommendation routes for the current request req.

State–action pair. For each state–action pair (sta_t, act_t), we need to extract corresponding features, denoted by $\phi(sta_t, act_t)$, based on which policy π_θ calculates the probability of taking action act_t. As in Section 4, for each possible action $act_t = r_q$, we extract route r_q's real-time internal and external features, i.e.,

$$\phi(sta_t, act_t = r_q) = f(r_q) = \langle f_{\text{int}}(r_q), f_{\text{ext}}(r_q) \rangle. \tag{11}$$

Policy. The policy $\pi_\theta(sta_t, act_t)$ is to output probability of taking each action $act_t = r_q \in \mathcal{R}(req)$, based on the extracted feature $f(r_q)$ of state–action pair $(sta_t, act_t = r_q)$.

Actually, $\pi_\theta(sta_t, act_t)$ is exactly the recommendation method introduced in Fig. 2. It first calculates the score $s(r_q) = h\left(f(r_q); \theta\right)$ for each possible action (route) $act_t = r_q$. Then, a softmax function is applied to obtain the probability of each action. That is, as Eq. (4) shows,

$$\pi_\theta(sta_t, act_t = r_q) = p(r_q) = \frac{\exp\left(h\left(f(r_q); \theta\right)\right)}{\sum_{q'=1}^{Q} \exp\left(h\left(f(r_{q'}); \theta\right)\right)}. \tag{12}$$

$Q = |\mathcal{R}(req)|$ is the number of candidate recommendation routes for the current request req, which could be different for different req.

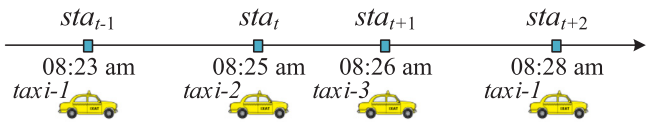


Fig. 8. State transition: an example.

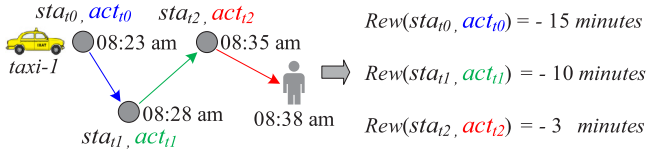


Fig. 9. Long-term reward: an example.

State transition. After a taxi takes an action act_t at state sta_t, the state evolves to next state sta_{t+1}, when there comes another taxi sending a new recommendation request. For example, as shown in Fig. 8, the current state is sta_t where taxi 2 sends a request for recommendation at 08:25 am. After taxi 2 takes an action, when next taxi (taxi 3) requires recommendation at 08:26 am, the state evolves to next state sta_{t+1}.

Reward. In this work, we directly define the long-term reward of a taxi taking action act_t at state sta_t, denoted by $Rew(sta_t, act_t)$. The long-term reward of a state–action pair will be used during the learning of policy π_θ. Formally, $Rew(sta_t, act_t)$ is defined as the time interval between now and the time when the taxi succeeds in picking up a passenger. In math, it is

$$Rew(sta_t, act_t) = -\left(time_{pickup} - time_t\right), \tag{13}$$

where $time_t$ denotes the time stamp of the current state sta_t and $time_{pickup}$ denotes the time stamp at which the taxi successfully picks up a passenger. For example, as shown in Fig. 9, taxi 1 takes action act_{t0} at state sta_{t0} at 08:23 am, action act_{t1} at state sta_{t1} at 08:28 am, action act_{t2} at state sta_{t2} at 08:35 am, and successfully picks up a passenger in route act_{t2} at 08:38 am. Then, we can obtain the long-term reward for each state–action pair, i.e., (see Fig. 9) $Rew(sta_{t0}, act_{t0}) = -15$, $Rew(sta_{t1}, act_{t1}) = -10$, and $Rew(sta_{t2}, act_{t2}) = -3$. It should be noted that usually $t1 \neq t0+1$. For the same taxi 1 in Fig. 8, $t0 = t - 1$ while $t1 = t + 2$.

5.2.2. Policy gradient

The objective of a reinforcement learning task is to learn an optimal policy π_θ (i.e., an optimal deep policy network) such that given any state sta, by taking actions following policy π_θ, taxis can obtain most long-term rewards (quickly pick up new passengers). Formally, it is

$$\max_\theta J(\theta) = \mathbb{E}_{sta\sim\pi_\theta}\left[\overline{Rew}(sta)\right], \tag{14}$$

where $\overline{Rew}(sta)$ denotes the *expected* long-term rewards that a vacant taxi can obtain at state sta by taking actions following policy π_θ [8]. $sta \sim \pi_\theta$ denotes sta is sampled by following π_θ.

To learn the optimal θ, we leverage a policy gradient algorithm called REINFORCE [42], which has been widely used in sequential decision-making problems [8,41]. The algorithm is demonstrated in Algorithm 1. The parameters θ in our policy network are first randomly initialized. Then, we continue updating θ until θ converges, i.e., the objective function $J(\theta)$ is maximized. The update process further consists of three steps.

First, Monte-Carlo sampling is used to sample state–action pairs (sta_t, act_t) and corresponding long-term rewards $Rew(sta_t, act_t)$. Specifically, given an initial state sta_0, we use policy π_θ (with the current θ) to select an action act_0. And then the state becomes sta_1 according to the environment, and likewise action act_1 is selected still using π_θ. The sampling is repeated until a predefined number T of state–action pairs and rewards are obtained.

Algorithm 1: Learning θ with REINFORCE

$\theta \leftarrow$ random initialization;
while θ *not converge* **do**
  MC-sampling: state–action pairs and rewards;
  calculate: gradient $\nabla_\theta J(\theta)$ (Eq. (15));
  update: $\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta)$;
end
return θ;

Second, we calculate the gradient of $J(\theta)$ to θ, i.e., $\nabla_\theta J(\theta)$. Based on [8,42], it is

$$\nabla_\theta J(\theta) = \mathbb{E}_{(sta,act)\sim\pi_\theta}\left[(\nabla_\theta \log \pi_\theta(sta, act)) \cdot \overline{Rew}(sta, act)\right],$$

where $\overline{Rew}(sta, act)$ denotes the *expected* long-term rewards of state–action pair (sta, act) by following policy π_θ [8]. $Rew(sta, act)$ is an specific instance of $\overline{Rew}(sta, act)$. Thus [42], using the sampled state–action pairs and rewards, $\nabla_\theta J(\theta)$ is

$$\nabla_\theta J(\theta) \approx \frac{1}{T}\sum_{t=0}^{T-1}\left[(\nabla_\theta \log \pi_\theta(sta_t, act_t)) \cdot Rew(sta_t, act_t)\right]. \tag{15}$$

Third, after having the gradient $\nabla_\theta J(\theta)$, we can update θ:

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta J(\theta), \tag{16}$$

where α is learning rate, e.g. $\alpha = 0.001$.

6. Evaluation

6.1. Dataset

To evaluate our dynamic taxi route recommendation method, we use open data in real world collected from San Francisco and New York. The taxi data used is publicly available from San Francisco [43] and New York [1]. The map data of the two cities is collected from OpenStreetMap [44].

6.1.1. Taxi data

San Francisco [43]. The taxi data in San Francisco contains GPS trajectory records of 536 taxis, collected from May 17, 2008 to June 10, 2008. In total, there exist 11,219,955 GPS points. Each GPS point is comprised of four elements (latitude, longitude, status, timestamp). The status denotes whether a taxi is occupied or vacant, from which we can extract the time stamps and locations of passengers being picked up and being dropped off. From taxis' GPS trajectories, we extract 414,381 passenger trips in the Bay Area of San Francisco.

New York [1]. New York has opened passengers' trip data. We collect 3 months of trip data from March 2015 to May 2015. Each passenger trip consists of much detailed information, e.g. the origin (latitude, longitude), destination (latitude, longitude), pickup time, drop-off time, and fare of a passenger. Thus we can also know passengers' information. In this work, we consider the trip data of yellow taxis in Manhattan, where we collect 32,638,971 passenger trips (354,771 trips per day).

6.1.2. Map data

The map data of New York and San Francisco are both obtained from OpenStreetMap [44]. For the Bay Area in San Francisco, after data cleaning and filtering, we obtain 3,087 vertices and 5,648 edges. For the Manhattan Area in New York, we obtain 5,118 vertices and 7,843 edges.

6.2. Experiment settings

In this work, we use the first 2/3 of data to learn the parameters of the deep policy network via deep reinforcement learning. The rest 1/3 of data is used to evaluate the learned policy network. Similar to prior literature [3,4,7,9,11,45], we also apply a simulated taxi system to do the evaluation. In our simulator, the time stamps and locations of coming passengers are identical to those in the collected real-world passenger trip data. For San Francisco, the number of taxis is set as 536 (same with the real-world dataset), while 10,000 taxis are used in New York. Taxis are randomly distributed initially. If a taxi is vacant, we will recommend a route for it. If the taxi does not find any waiting passengers after reaching the end of the recommended route, we will recommend a new route to it. The travel time of taxis is given by the simulator, based on the travel distance and the speed limit of each road segment. Our method is realized in C++ (for simulation) and Python (for learning deep policy network). Experiments are conducted on a server with 2.60 GHz Intel(R) Xeon(R) CPU and 126 GB.

6.3. Baselines

To evaluate our dynamic taxi route recommendation method, we compare it with the following baseline methods.

- **MPP** [3]. This method recommends a route with the maximum pickup probability to a vacant taxi. The pickup probability of each edge is estimated using the historical average, not the real-time pickup probability of an edge.
- **MNP** [7]. This method recommends a route with the maximum net profit, where similarly the net profit of an edge is also obtained from historical data.
- **PCD** [10]. This method recommends a route with the minimal potential cruising distance. Each candidate recommendation route's potential cruising distance is learned from historical data, too.
- **MDM** [9]. This method recommends a route aiming at minimizing the vacant cruising distance of a taxi, using Monte Carlo tree search. MDM [9] is the most state-of-the-art.
- **TQL** [11]. This method provides a table Q-learning method to do the taxi route recommendation. Monte Carlo sampling methods are used to learn the table q-value of each state–action pair [8].

6.4. Metrics

Following the metric proposed in [9], we also use the improvement ratio of our recommendation method over baseline methods as our evaluation metrics. The difference is that we consider more metrics to do the evaluation more carefully. Specifically, we apply the following four metrics.

Improvement ratio of taxi drivers' average earning. For each baseline method (denoted by *basl*), the improvement of taxi drivers' earnings using our deep reinforcement learning-based recommendation method (denoted by *DRL*) over *basl* is defined as

$$Improve_{basl}^{Earn} = \frac{Earn_{DRL} - Earn_{basl}}{Earn_{basl}}. \tag{17}$$

$Earn_{DRL}$ and $Earn_{basl}$ denote the taxi drivers' average earnings using our method and each baseline method $basl \in$ [MPP, MNP, MDM, QL], respectively.

Improvement ratio of taxis' average vacant cruising time:

$$Improve_{basl}^{VCT} = \frac{VCT_{basl} - VCT_{DRL}}{VCT_{basl}}. \tag{18}$$

Table 1
Improvement ratios of our method over baselines. Our method: $l = 4$, $k = 2$.

Method	SF dataset				NY dataset			
	Earn	VCT	WT	#PP30	Earn	VCT	WT	#PP30
MPP	2.238	0.177	0.548	2.479	1.303	0.226	0.727	1.310
MNP	0.581	0.105	0.444	0.620	1.030	0.207	0.714	1.040
PCD	2.017	0.171	0.544	2.162	1.162	0.217	0.722	1.163
MDM	5.549	0.204	0.556	5.966	0.428	0.137	0.634	0.427
TQL	0.344	0.067	0.296	0.300	0.158	0.068	0.491	0.159

VCT_{DRL} and VCT_{basl} denote taxis' vacant cruising time using our method and baseline method *basl*, respectively.

Improvement ratio of passengers' average waiting time:

$$Improve_{basl}^{WT} = \frac{WT_{basl} - WT_{DRL}}{WT_{basl}}. \tag{19}$$

WT_{DRL} and WT_{basl} denote passengers' average waiting time using our method and baseline method *basl*, respectively.

Improvement ratio of passengers being picked up in 30 minutes:

$$Improve_{basl}^{\#PP30} = \frac{\#PP30_{basl} - \#PP30_{DRL}}{\#PP30_{basl}}. \tag{20}$$

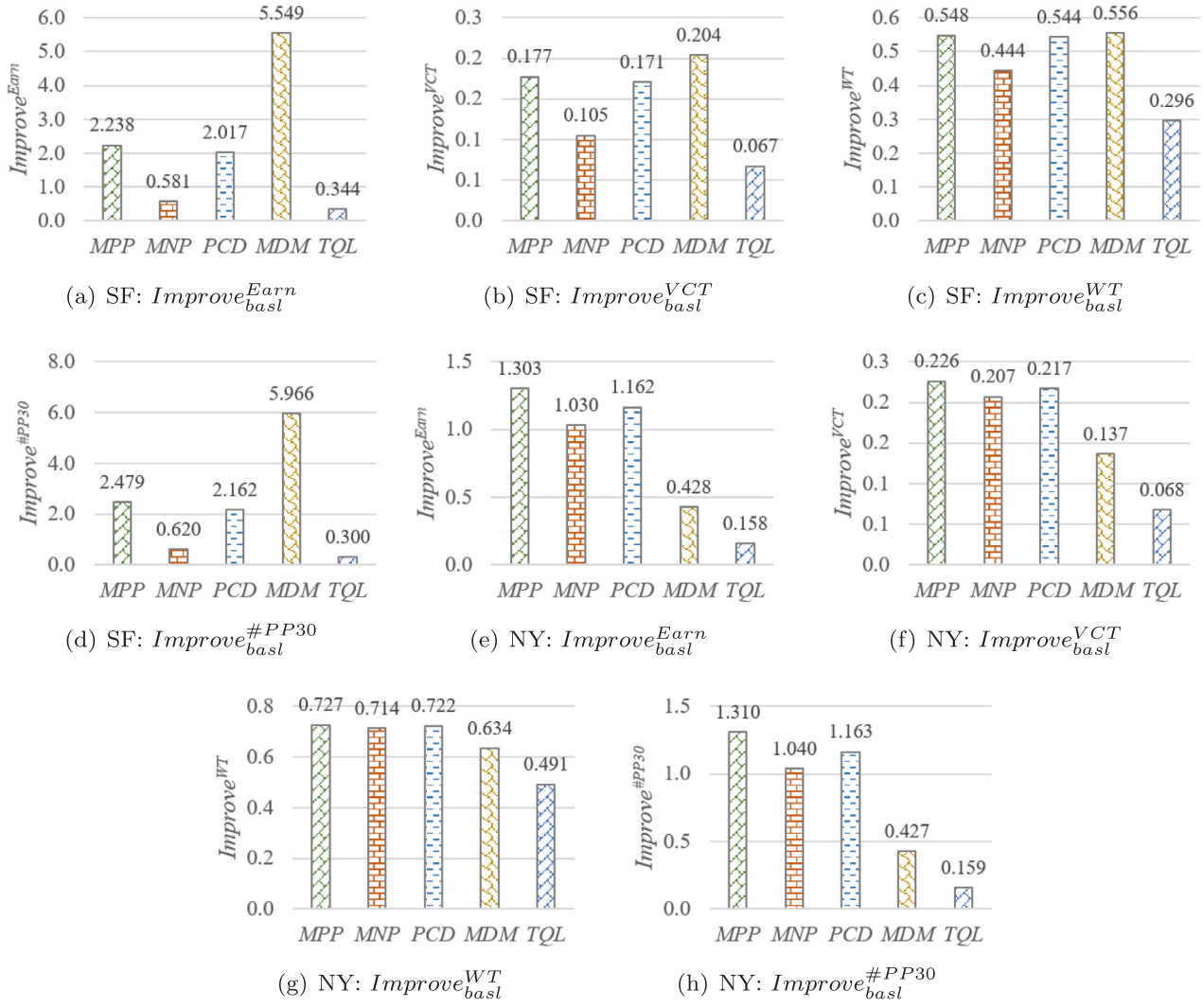
$\#PP30_{DRL}$ and $\#PP30_{basl}$ denote the number of passengers being picked up in 30 min using our method and baseline method *basl*, respectively.

6.5. Effectiveness

The improvement ratios of our taxi route recommendation method over baseline methods are presented in Fig. 10 and Table 1. As shown in Figs. 10(a) and 10(e), the improvement ratios of taxi drivers' average earning using our method over baselines are at least 34.4% (over TQL) and 15.8% (over TQL) in San Francisco and New York, respectively. In terms of taxis' average vacant cruising time (see Figs. 10(b) and 10(f)), our method can reduce at least 6.7% and 6.8% of average vacant cruising time for taxis in two cities, respectively. Similarly, for passengers, our method can reduce at least 29.6% and 49.1% of average waiting time for each passenger as shown in Figs. 10(c) and 10(g). The ratio of passengers being picked up in 30 min is also significantly improved, i.e., 30% and 15.9% in Figs. 10(d) and 10(h), respectively. Based on these results, we can safely conclude that our method can significantly improve the transportation efficiency of taxis, increasing taxi drivers' earnings and reducing passengers' waiting time. The good performance of our method can be attributed to two reasons. First, just as discussed in Introduction, we consider more realistic and useful features, i.e., the real-time internal and external spatio-temporal features. Second, to better fuse these features, we design a deep policy network learned by deep reinforcement learning.

6.6. Comparisons between different deep reinforcement learning methods

In this work, the proposed deep reinforcement learning method is a REINFORCE policy-based method (see Algorithm 1). To demonstrate its performance, we compare it with the other two deep reinforcement learning methods, Proximal Policy Optimization (PPO) [27] and Deep Q-Learning (DQN) [23,24]. PPO is also a policy-based method, while DQN is a value-based method [8]. To make fair comparisons, the settings of the PPO and DQN are the same with the setting of our REINFORCE. Specifically, we use the same features (i.e., the real-time internal and external spatio-temporal features) for both PPO and DQN. The network structures of the PPO and DQN are also the one in Fig. 6. The



(a) SF: $Improve^{Earn}_{basl}$ (b) SF: $Improve^{VCT}_{basl}$ (c) SF: $Improve^{WT}_{basl}$

(d) SF: $Improve^{\#PP30}_{basl}$ (e) NY: $Improve^{Earn}_{basl}$ (f) NY: $Improve^{VCT}_{basl}$

(g) NY: $Improve^{WT}_{basl}$ (h) NY: $Improve^{\#PP30}_{basl}$

Fig. 10. The improvement ratios of our method over baseline methods. Our method: $l = 4$, $k = 2$.

Table 2
The improvement ratios of REINFORCE, PPO and DQN over baseline TQL.

Method	SF dataset				NY dataset			
	Earn	VCT	WT	#PP30	Earn	VCT	WT	#PP30
REINFORCE	0.344	0.067	0.296	0.300	0.158	0.068	0.491	0.159
PPO	0.333	0.066	0.312	0.302	0.158	0.068	0.457	0.158
DQN	0.194	0.032	0.178	0.151	0.022	0.009	0.053	0.023

learning rates for PPO and DQN are also set as 0.001. The Monte Carlo sampling method is also used in the DQN, in which the Q-network is to estimate the long-term reward of each state–action pair. Experimental results are demonstrated in Fig. 11 and Table 2.

As shown in the results, both the REINFORCE and the PPO have significant improvement ratios over the baseline TQL method. Note that the TQL method performs the best among all baselines (see Table 1), thus we calculate the improvement ratios over the TQL. Besides, the REINFORCE and the PPO achieve much better performance than the DQN. Thus, the policy-based RL method is more suitable than the value-based RL method in this taxi route recommendation problem. This could be due to that the taxi system has high uncertainty and randomness, such as the coming passengers (time stamps and locations), the traffic conditions, and so on. It makes the Q-network in the DQN hard to accurately predict the long-term reward of an action. In addition, many actions may have near-maximal long-term rewards, making the prediction more difficult. These make the DQN cannot perform well in the taxi route recommendation problem. In fact, existing studies have shown that DQN performs poorly even in simple problems [27,46,47].

6.7. Discussion on parameters l and k

Fig. 12(a) presents the performance of our method under different recommendation route lengths l. The experiment is conducted in SF data and we demonstrate the improvement ratios of our method over TQL method. Clearly, with the increase of recommendation route length l, the performance of our method also increases. This is due to that with a longer recommendation route, more candidate routes will be generated, and our method is more likely to recommend a better route to a vacant taxi.

Likewise, Fig. 12(b) discusses the situation where our method considers different orders of surrounding grids k. As the figure demonstrates, considering more orders of surrounding grids results in better recommendation results. This is due to that with more orders of surrounding grids under consideration, our method can know a route's pickup easiness in a longer future.

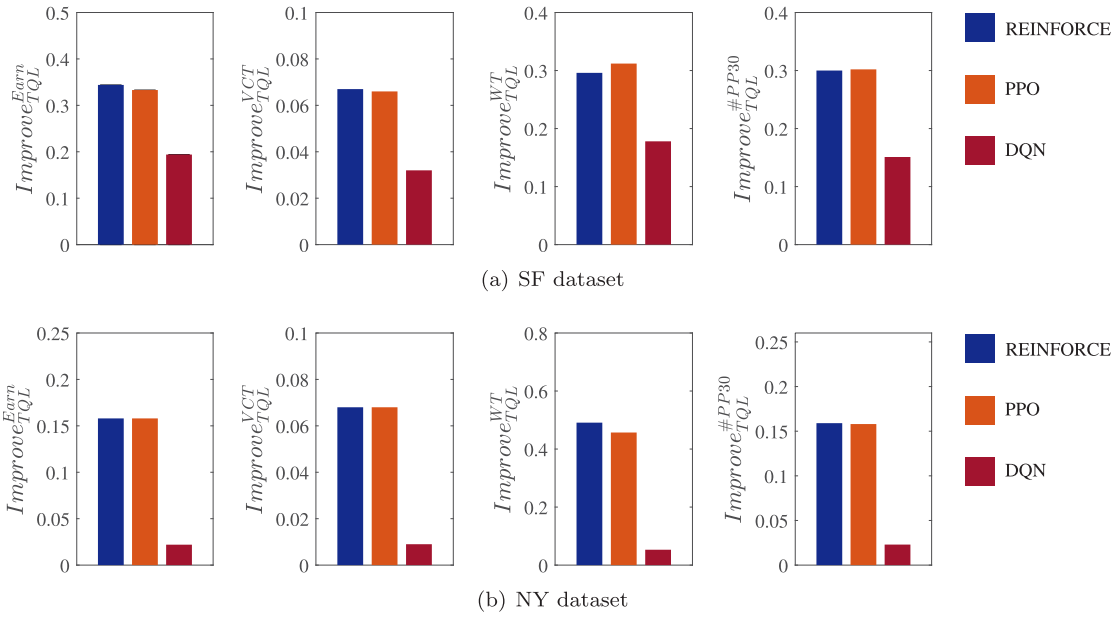


Fig. 11. Improvement ratios of REINFORCE, PPO and DQN over baseline TQL.

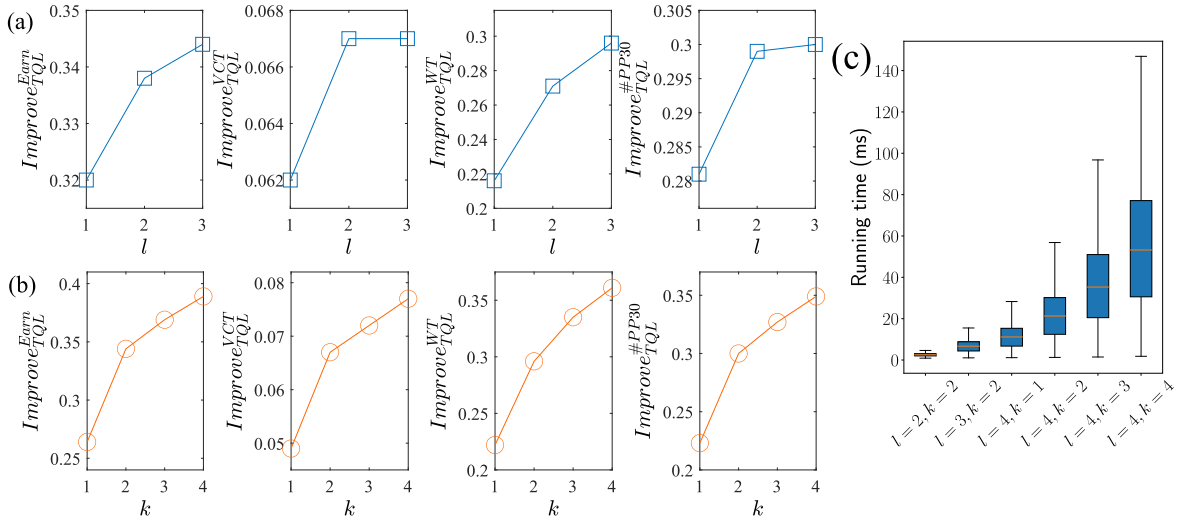


Fig. 12. (a) Different recommendation route lengths l. $k = 2$. (b) Considering different orders of surrounding grids k. $l = 4$. (c) Running time of our method with different k and l (in milliseconds).

6.8. Running time analysis

The box plot in Fig. 12(c) presents the running time needed for our method to conduct one route recommendation for a vacant taxi. It shows that our method spends less than 57 ms on average for all settings. With the increases of l and k, our method requires more running time, since for bigger l and k there are more candidate recommendation routes to consider and more features to extract. Fortunately, even under $l = 4$ and $k = 4$, our method is still highly efficient (using ∼57 ms on average and ∼148 ms at most). Therefore, with the larger l and k, on one hand the performance of our method will be better, on the other hand the running will become slower. Fortunately, our method can easily run in parallel, since each candidate recommendation route's score can be computed independently (Fig. 6) and thus can be computed in parallel. Hence, the more computing resources that a taxi system owner has, the larger l and k can be set, and the better performance can be obtained. In fact, even when l and k are small (e.g. l = 4 and k = 2) and without parallel computing, the performance of our method has already largely defeated baselines (Fig. 10), and the running time is undoubtedly acceptable (Fig. 12(c)). Thus, our method is highly efficient and taxi system owners can select proper parameters for l and k based on the computing resources they have and the performance they expect.

7. Conclusion and future work

In this paper, we designed an adaptive deep reinforcement learning method to fuse the spatio-temporal features to do the dynamic taxi route recommendation. Real-time internal and external spatio-temporal features are formally formulated and extracted. With our method, the transportation efficiency of taxis can be significantly improved. Specifically, comparing with the state-of-the-arts, our method is able to reduce at least 29.6% of average waiting time for passengers and increase at least 15.8% of average earning for taxi drivers. In the future, we are interested in the joint optimization of taxi route recommendation,

taxi dispatching, and taxi sharing, such that taxis' transportation efficiency can be jointly improved. In particular, we expect to study how to use deep reinforcement learning to do the joint optimization.

CRediT authorship contribution statement

Shenggong Ji: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Zhaoyuan Wang:** Methodology, Writing - review & editing. **Tianrui Li:** Supervision, Conceptualization, Writing - review & editing, Funding acquisition. **Yu Zheng:** Supervision, Conceptualization, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the National Key R&D Program of China (No. 2019YFB2101801) and the National Natural Science Foundation of China (No. 61773324).

References

[1] B. Donovan, D.B. Work, New York City Taxi Trip Data (2010-2013), 2016.
[2] J. Yu, P. Lu, Learning traffic signal phase and timing information from low-sampling rate taxi GPS trajectories, Knowl.-Based Syst. 110 (2016) 275–292.
[3] J. Yuan, Y. Zheng, L. Zhang, X. Xie, G. Sun, Where to find my next passenger, in: UbiComp 2011: Ubiquitous Computing, 13th International Conference, ACM Press, New York, NY, 2011, pp. 109–118.
[4] H. Rong, X. Zhou, C. Yang, M.Z. Shafiq, A.X. Liu, The rich and the poor: A Markov decision process approach to optimizing taxi driver revenue efficiency, in: Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, in, USA, October 24–28, 2016, pp. 2329–2334.
[5] R. Wang, C. Chow, Y. Lyu, V.C.S. Lee, S. Kwong, Y. Li, J. Zeng, TaxiRec: Recommending road clusters to taxi drivers using ranking-based extreme learning machines, IEEE Trans. Knowl. Data Eng. 30 (3) (2018) 585–598.
[6] Y. Zheng, L. Capra, O. Wolfson, H. Yang, Urban computing: concepts, methodologies, and applications, ACM Trans. Intell. Syst. Technol. 5 (3) (2014) 38:1–38:55.
[7] M. Qu, H. Zhu, J. Liu, G. Liu, H. Xiong, A cost-effective recommender system for taxi drivers, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, ACM Press, New York, NY, 2014, pp. 45–54.
[8] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, second ed., MIT Press, Cambridge, MA, 2017, in progress.
[9] N. Garg, S. Ranu, Route recommendations for idle taxi drivers: Find me the shortest route to a customer!, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, ACM Press, New York, NY, 2018, pp. 1425–1434.
[10] Z. Luo, H. Lv, F. Fang, Y. Zhao, Y. Liu, X. Xiang, X. Yuan, Dynamic taxi service planning by minimizing cruising distance without passengers, IEEE Access 6 (2018) 70005–70016.
[11] T. Verma, P. Varakantham, S. Kraus, H.C. Lau, Augmenting decisions of taxi drivers through reinforcement learning for improving revenues, in: Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling, ICAPS 2017, Pittsburgh, Pennsylvania, USA, June 18-23, 2017, pp. 409–418.
[12] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, J. Ye, A taxi order dispatch model based on combinatorial optimization, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017, pp. 2151–2159.
[13] Z. Xu, Z. Li, Q. Guan, D. Zhang, Q. Li, J. Nan, C. Liu, W. Bian, J. Ye, Large-scale order dispatch in on-demand ride-hailing platforms: a learning and planning approach, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, pp. 905–913.
[14] R. Zhang, M. Pavone, Control of robotic mobility-on-demand systems: A queueing-theoretical perspective, I. J. Robotics Res. 35 (1–3) (2016) 186–203.
[15] S. Ma, Y. Zheng, O. Wolfson, T-share: A large-scale dynamic taxi ridesharing service, in: 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013, pp. 410–421.
[16] D.O. Santos, E.C. Xavier, Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem, in: IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, pp. 2885–2891.
[17] K. Lin, R. Zhao, Z. Xu, J. Zhou, Efficient large-scale fleet management via multi-agent deep reinforcement learning, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, 2018, pp. 1774–1783.
[18] W. Lee, S. Tseng, J. Shieh, H. Chen, Discovering traffic bottlenecks in an urban network by spatiotemporal data mining on location-based services, IEEE Trans. Intell. Transp. Syst. 12 (4) (2011) 1047–1056.
[19] M. Pan, W. Huang, Y. Li, X. Zhou, Z. Liu, R. Song, H. Lu, Z. Tian, J. Luo, DHPA: Dynamic human preference analytics framework: A case study on taxi drivers' learning curve analysis, ACM Trans. Intell. Syst. Technol. 11 (1) (2020).
[20] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting, CoRR abs/1709.04875 (2017).
[21] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, URL http://www.deeplearningbook.org.
[22] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[23] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (2017) 26–38.
[24] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[25] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016.
[26] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, N. De Freitas, Dueling network architectures for deep reinforcement learning, 2015, arXiv preprint arXiv:1511.06581.
[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[28] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T.P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabi, Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489.
[29] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of go without human knowledge, Nature 550 (2017) 354–359.
[30] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (2016) 39:1–39:40.
[31] H. Tang, Y. Wang, X. Liu, X. Feng, Reinforcement learning approach for optimal control of multiple electric locomotives in a heavy-haul freight train:A double-switch-Q-network architecture, Knowl.-Based Syst. 190 (2020) 105173.
[32] C. Martinez, E. Ramasso, G. Perrin, M. Rombaut, Adaptive early classification of temporal sequences using deep reinforcement learning, Knowl.-Based Syst. 190 (2020) 105290.
[33] S. Ji, Y. Zheng, Z. Wang, T. Li, A deep reinforcement learning-enabled dynamic redeployment system for mobile ambulances, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 3 (1) (2019).
[34] G. Sun, D. Ayepah-Mensah, A. Budkevich, G. Liu, W. Jiang, Autonomous cell activation for energy saving in cloud-RANs based on dueling deep Q-network, Knowl.-Based Syst. 192 (2020) 105347.
[35] S.S. Skiena, The Algorithm Design Manual: Text, vol. 1, Springer Science & Business Media, 1998.
[36] J. Durbin, S.J. Koopman, Time Series Analysis by State Space Methods, second ed., 2012.
[37] Y. Tong, Y. Chen, Z. Zhou, L. Chen, J. Wang, Q. Yang, J. Ye, W. Lv, The simpler the better: A unified approach to predicting original taxi demands based on large-scale online platforms, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13–17, 2017, pp. 1653–1662.
[38] J. Xu, R. Rahmatizadeh, L. Bölöni, D. Turgut, Real-time prediction of taxi demand using recurrent neural networks, IEEE Trans. Intell. Transp. Syst. 19 (8) (2018) 2572–2581.
[39] L. Zhao, Y. Zhou, H. Lu, H. Fujita, Parallel computing method of deep belief networks and its application to traffic flow prediction, Knowl.-Based Syst. 163 (2019) 972–987.

[40] L. Li, L. Qin, X. Qu, J. Zhang, Y. Wang, B. Ran, Day-ahead traffic flow forecasting based on a deep belief network optimized by the multi-objective particle swarm algorithm, Knowl.-Based Syst. 172 (2019) 1–14.
[41] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, J. Artificial Intelligence Res. 4 (1996) 237–285.
[42] R.S. Sutton, A.G. Barto, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (1992) 229–256.
[43] M. Piorkowski, N. Sarafijanovoc-Djukic, M. Grossglauser, A parsimonious model of mobile partitioned networks with clustering, in: The First International Conference on COMmunication Systems and NETworkS (COMSNETS), 2009.
[44] OpenStreetMap contributors, 2017, Planet dump retrieved from https://planet.osm.org.
[45] R. Wang, C. Chow, Y. Lyu, V.C.S. Lee, S. Kwong, Y. Li, J. Zeng, TaxiRec: Recommending road clusters to taxi drivers using ranking-based extreme learning machines, IEEE Trans. Knowl. Data Eng. 30 (3) (2018) 585–598.
[46] Y. Duan, X. Chen, R. Houthooft, J. Schulman, P. Abbeel, Benchmarking deep reinforcement learning for continuous control, CoRR abs/1604.06778 (2016) arXiv:1604.06778.
[47] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, 2016, arXiv:arXiv:1606.01540.