A Deep Q-Learning Network Based Reinforcement Strategy for Smart City Taxi Cruising

Zhenyao Hua[1,2,3], Dongyang Li[1,3], and Weian Guo[1,2,3](✉)

[1] School of Information and Electronics Engineering, Tongji University, Shanghai 201804, China
{2032976,guoweian}@tongji.edu.cn

[2] Sino-German College of Applied Sciences, Tongji University, Shanghai 201804, China

[3] Shanghai Institute of Intelligent Science and Technology, Tongji University, Shanghai 200092, China

Abstract. Smart transportation is crucial to citizens' living experience. A high efficiency dispatching system will not only help drivers rise income, but also save the waiting time for passengers. However, drivers' experience-based cruising strategy cannot meet the requirement. By conventional strategies, it is not easy for taxi drivers to find passengers efficiently and will also result in a waste of time and fuel. To address this problem, we construct a model for taxi cruising and taking passengers based on the view of drivers' benefits. By employing real data of taxi orders, we apply a deep-Q-network in the framework of reinforcement learning to find a strategy to reduce the cost in taxi drivers' finding the passengers and improve their earning. Finally, we prove the effect of our strategy by comparing it with a random-walk strategy in different segments of time both in workday and weekend.

Keywords: Taxi cruising strategy · Deep Q-Network · Reinforcement learning · Drivers' benefits

1 Introduction

As an important role in urban transportation network, taxis provide convenient and comfortable services for citizens travelling. However, there still exist many problems in the supply and demand of service between taxis and clients, resulting in the fact that many taxis stay vacant in a large percent of their working time while some clients should wait for a long time to take a taxi. Such cases will cause a large waste of energies and time from both drivers and clients [1]. With the development of information technology, it is possible to apply artificial intelligence in taxis cruising to enhance the service efficiency. In early 1990s, Dial proposed a phone-calling-based system for taxi service [2]. Currently, thanks to the appearance of smart phones and GPS technology, it is very popular to use APPs for booking and calling taxis service. Consequently, lots of transportation network companies appear, such as Didi, Uber and many others. By

H. Zhang et al. (Eds.): NCAA 2021, CCIS 1449, pp. 59–70, 2021.
https://doi.org/10.1007/978-981-16-5188-5_5

building an information platform, both locations of taxi drivers and clients are shared on it to call and response taxis services [3].

In general, taxis cruising strategy much influences drivers' income and clients' waiting time. When a taxi is non-hailing, drivers usually cruise randomly or go to a target area according to drivers' own experiences. However, an inferior taxi dispatching strategy will cause increasing non-hailing time, more costs for taxi drivers and result in a long waiting time for clients [4]. An intelligent and optimized taxi dispatching system will much enhance the efficiency of taxi service. For such issue, during past decades, a few studies tried to discover the pattern of the human drivers' real actions based on taxis trajectories [5–7]. Some researchers used the real time information of the taxis distribution in a city to evaluate the saturation degree of different regions and then schedule the taxis [8–10]. Liu constructed the scheduling model based on three aspects involving clients demands, taxis distribution and routing situation by comparing both real-time data and historical data [11]. Swarm intelligence was also used to realize a more sensible global resources allocation [12, 13]. Markov decision process [14, 15] and Q-learning [16, 17] were investigated in similar studies. However, the current researches focus on the macroscopic view, but lacks of related researches from taxi agents' standpoint for cruising strategy.

To address this problem, in this paper, we contribute the following works. We employ a deep Q-network (DQN) and taxi trajectory data to train a taxi cruising model in order to provide a strategy for the taxi drivers when non-hailing. Based on such strategy, the drivers will have a large probability to increase their income and reduce the waste of both time and fuel.

The remainder of this paper is organized as follows. In Sect. 2, we construct a model for taxi cruising based on the view of drivers' benefits. Section 3 introduces a Q-network in detail and describes the DQN learning procedures. Section 4 conducts experiments and simulations to evaluate the proposed models in different time zones in one day. The proposed strategy is also analyzed and compared with a random-walk strategy. We end this paper in Sect. 5, and present our future work.

2 Problem Description

2.1 Modeling

In general, taxi drivers cruise based on their own experience, for different kinds of purposes, such as total profit or total accounts of service, etc. In addition, some other factors in the real scenarios are also supposed to be considered, which involve the position and total work time of the taxi driver. There also exists uncertainties in clients demands, because in different periods of one day or in different dates, the transportation situations are not similar. Besides, the weather conditions and citizens preferences will also affect the transportation situation so that it is difficult to build an analytical formula for taxis dispatching system just by drivers' experience. Therefore, it is necessary to apply artificial intelligence on such scenarios. In this paper, reinforcement learning is employed to address the problems. In the framework of reinforcement learning, we take taxis as agents, while the taxi service environment is regarded as a learning environment.

The objective of a cruising strategy, shown in (1), is modeled as an optimization problem which maximizes drivers' profit.

$$Maximize : P = \sum_i G_i - \sum_j F_j \tag{1}$$

where P is the total profit of a period in one period of work, G_i is the taken income in the ith order, while F_j is the comsumption of oil fee in the jth driving. In general, a driver will stay in three states which are cruising, picking and carrying. For the cruising state, the driver is cruising and finding customers. Once the driver responses an order, the driver will go into picking state to pick up the clients. During picking state, there is no earning. After the clients get on the taxi, the taximeter begins to count and the driver will be in carrying state.

There are three states defined in the proposed model, which are cruising, picking and carrying. A taxi is initialized in the state of cruising. It always transits through the picking state to the carrying state. When the carrying state is over, it will check whether there is an order immediately to determine whether it transits back to the cruising state or to the picking state (the pattern after the F_3 and F_7). In the Fig. 1, all the three states will cost fees and only the carrying state will bring income.

A typical pattern of the state transition is shown in Fig. 1 from t_0 to t_3. The taxi starts with the cruising state during t_0 and t_1. When it finds an order, the state changes to picking state at t_1 and the state lasts to t_2. When the taxi picks up a passenger at t_2, the state changes to carrying state. Finally, when the taxi finishes an order task at t_3, the state changes back to the cruising. In the bottom two lines in Fig. 1, all the three states will cost fees and only the carrying state will bring income.

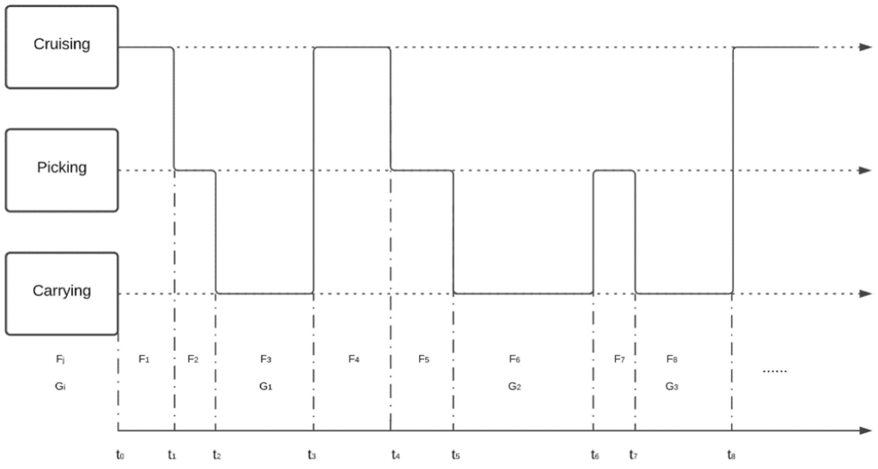

Fig. 1. The picture is an example illustrating the possible states transition of a taxi as time passes.

Considering that a taxi cannot work too much time due to the limitation of fuel, or the service policy of a driver, we split one day into several segments. In addition, for the learning environment, a whole city is divided by N in both north-south direction and east-west direction into grids. Namely, the city area is divided into N * N grids. When a taxi transits to the cruising state it uses the grid index as the function input to determine which grid it goes to, otherwise it is more difficult and consumed in computation. Therefore, the strategy can be defined as follows.

$$T = S(x,y) \tag{2}$$

where S is the strategy function and T is the strategy target when the taxi is in the grid of (x, y). The *T* is also a two-dimension grid index. Figure 2 is an example where the taxi is in grid (0, 0) and the strategy is to go to grid (2, 2). In a period of time the strategy is the same, and in different periods of time we trained different strategy functions.

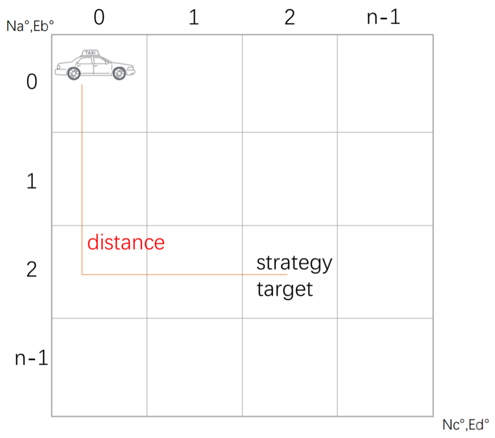

Fig. 2. The picture shows how the considered region is divided into grids and the input and output of the strategy in our model. The distance of two positions is calculated in the form of Manhattan Distance.

In the modelling, some preliminaries and assumptions are made as follows.

1. For a booking order, it includes the GPS position of start position and destination position, the booking time and the price.
2. For a booking order, the transportation process is ignored. The distance travelled is calculated according to Manhattan Distance as shown in Fig. 2.
3. For a grid which contains more than one taxi, they have competition relationship. Only one taxi can response an order with a predefined probability.
4. From start position to destination position, a taxi will not response to any order during the transportation process.

2.2 Brief of Deep Reinforcement Learning

Reinforcement learning (RL), as a powerful tool in machine learning, has been well applied to various implementations, such as workflow scheduling [18], automated radiation adaption in lung cancer [19], stock market forecasting [20], etc. In RL, by iterative interactions with environments, agents will learn an optimized policy for guiding their behaviors. For each interaction, agents are expected to make a smart decision to pursue more rewards [21]. At each time step t, an agent is described by a state s_t from state-space, while it will also select a feasible action a_t from the action-space, following a policy $\pi(a_t|s_t)$. Then the agent receives a reward r_t and transits to the next state s_{t+1}. The process repeats until a predefined termination condition is met. The goal of reinforcement learning is to obtain a policy $\pi(a_t|s_t)$ to maximize the accumulated rewards with a discount factor $\gamma \in [0, 1]$. The accumulated reward is defined in (3):

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{3}$$

where R_t is the accumulated rewards at the t time step, γ is the discount factor and r_{t+k} is the expected reward got at the $t + k$ time step. In this paper, Q-network learning [22] is employed to optimize the policy π. In Q-network, Q is used to depict action value and defined in (4):

$$Q_\pi(s, a) = E(R_t|s_t = s, a_t = a) \tag{4}$$

where the right part is the expected accumulated reward for the selected action a in state s. Then we can get a policy in (5):

$$\pi(s) = \underset{a}{argmax}(Q_\pi(s, a)) \tag{5}$$

where $\pi(s)$ is the policy function to get the selected action a by solving a maximization function. Namely the action a will help the agent get the largest expected accumulated reward in the current Q function. During the process, we employ temporal difference (TD) learning strategy with bootstrapping, as a kernel part, in reinforcement learning to optimize Q function from agents' experience. In this way, the whole learning process will be model-free, online, and fully incremental. According to one time of experience, (st, at, st + 1, rt), the TD error is

$$E_{td} = \left(r_t + \gamma \max_a Q(s_{t+1}, a)\right) - Q(s_t, a_t) \tag{6}$$

and the update rule is

$$Q(s, a) \leftarrow Q(s, a) + \eta E_{td} \tag{7}$$

where η is the learning rate. Q-learning converges to the optimum action-values with probability 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely[22].

Considering that there are thousands of grids for a city and the scale of (state, value) pairs are too large, it is not feasible to build a Q-table. Here we use a neural network to represent the Q function and call this deep reinforcement learning. The detail of the network design is introduced in Sect. 3. The input of the neural network is the state and output is the Q-value of every action. The parameters in the neural network is updated through gradient descent as the update rule mentioned above. The Q function may not stay stable when the function is in the form of approximation [23, 24]. To stabilize the learning process, there are still many essential operations [25] like replacing buffer, target network and hyperparameter tuning. To balance the exploration and exploitation, epsilon greedy is used in this paper.

3 Design of DQN

3.1 Network Expressed Strategy

As explained in Sect. 2, the cruising strategy obtained in this paper decides the next position only depends on which grid the taxi is in. We use a neural network to express the strategy and it is also the action value function Q, whose input is the index of grid where the taxi is. The output of neural network is the Q value of every action. Therefore, we can get the strategy output according to formula (3). We do not use the grid number but use a two-dimension index as the input because the adjacent grid may have a consistency of the regular pattern. The structure of the network is shown in Fig. 3.

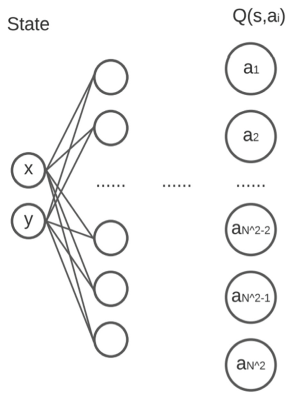

Fig. 3. The picture shows the structure of the network. The leftmost two nodes are the input nodes while the rightmost row of nodes are the output nodes, and others are intermediate nodes.

3.2 Procedure

In the model training procedure, though generally there are many steps in a normal single episode, in this paper every episode has only one step. When the episode begins, the taxi is initialized in a random location in the map with a random working time. Initially, for each taxi, there is no passengers on it, and then the taxi will take an action toward another location block. After that, the environment, namely the dispatching system, will check whether there is an order in that location at that time. If there is not an order, the episode is over. If there exist some orders, the taxi will take one of them and implement the order to move to the destination position. After finish the task, a taxi will repeat cruising, picking and carrying states.

The strategy still aims for the total profit for a whole taxi work time because the temporal difference learning is used and the Q value of the next state is considered. Every Q value means the expected profit at an action for a state in an infinite sequence restrained by the discount factor γ. The episode will be repeated for many times, so the agent may learn much experience from the specific time period.

Figure 4 shows information flow in the training procedure. A taxi interacts with the environment with the information of the orders and maps. The interaction information stored in the replay buffer for further trains. A strategy will be trained iteratively and used in the future policy decision for a taxi.

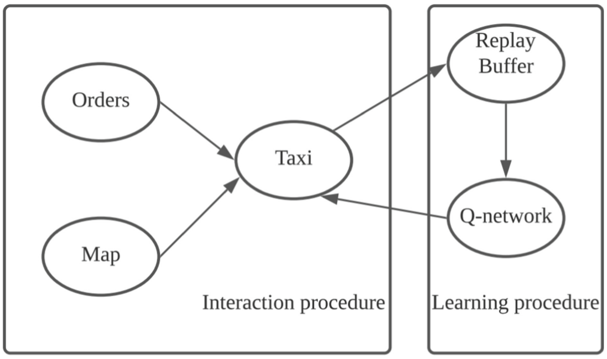

Fig. 4. The picture shows the information flow in the training procedure.

For Q-learning process, state, action and reward are depicted as follows.

1) State: We only use a location to establish the state of a taxi. More precisely, the state of the taxi is of two degrees, the segment number in the east-west direction and in the south-north direction, as shown in Fig. 2. Both of them are integers in the range of 0 to N-1.
2) Action: As described in Sect. 2, an action instructed by the Q network for the taxi is another position described by the block number. For the simplicity, it is the same as the state as a two-dimension integer tuple.
3) Reward: Whenever the taxi runs, it results in a minus reward as a cost of oil. It is calculated by multiplying a factor on the distance. Whenever the taxi takes an order, it results in a positive reward.

The procedure to train a Q-network suitable for a different time is only different in the initialized time of the taxi when the episode begins.

Algorithm 1 provides the pseudo-codes for the learning process, which follows DQN learning process and is adapted for our requirements in the part of interaction with the environment.

4 Experiments and Results

In this section, we conduct experiments to validate the performance of the proposed algorithm. The data is obtained from GAIA Open Dataset provided by DiDi company, which depicts taxis operations in Chengdu City. The dataset includes booking orders in November 2016, which records the cryptographic and anonymous location data of order time, start position and destination position. To eliminate the sporadic data, we sample the data in Chengdu from the west-south corner named Tianfufurong Garden to the east-north corner named Qilong Temple. Based on the information, we calculated the payment price of each order by Ali cloud.

We divide the whole map into 100 grids by N with size of 10. The Q-network is a mapping from a state to the Q value of every action, so the number of input nodes is two and the number of the output nodes is 100. We design the number of hidden layer as 3. The number of nodes respectively is set as 20, 80, 160 respectively.

In this experiment, a taxi works for a period of four hours. In the following tables we list four kinds of indices, the total income of the four hours, the total time used to pick up the passengers, the total non-hiring time, and the number of the total orders. We test three models separately trained for three periods of time from 8:00 to 12:00, from 16:00 to 20:00 and from 20:00 to 24:00. For each time period, we conduct the test for 20 times and record the mean value (AVG), the standard deviation (SD).

Regardless of whether it is workday or weekend, we present the comparison results in Table 1. It is obvious that the proposed algorithm marked by "DQN" has much better performance than the random-walk strategy marked by "rand". For the index of income and order number, DQN has improved almost 350% and 220% respectively. The increasement of pick time is relevant to the order number. The non-hiring time also reduces a lot. Besides, the standard deviation reduces significantly only except the order number and pick time during the period of 20:00 to 24:00.

Table 1. Training effect of non-distinguishing workday and weekend compared with a random strategy in three periods of time

		Income		Pick time		Vacant run time		Order num	
		AVG	SD	AVG	SD	AVG	SD	AVG	SD
8:00–12:00	rand	40.68	52.17	722.07	552.0	9842.2	3419.6	4.8	3.19
	DQN	141.79	24.80	1570.7	367.8	3588.1	1298.6	10.6	2.08
16:00–20:00	rand	39.96	56.65	741.64	537.0	9908.7	3480.9	4.55	3.29
	DQN	140.81	22.02	1638.7	417.8	3899.9	1201.7	10.7	2.02
20:00–24:00	rand	22.97	45.67	520.52	356.0	11549	2899.8	3.6	2.43
	DQN	123.20	38.90	1083.3	447.9	4878.9	2730.3	7.8	2.96

Then we train our model specialized only for workday and weekend. We test for corresponding date with the specialized models and list the same kinds of indices respectively in Tables 2 and 3 respectively. The increasement of income are also remarkable and the standard deviations in "DQN" algorithm are smaller than the "rand" strategy. The difference from the results in Table 1 are mainly displayed in the decrease in the standard deviation.

Table 2. Training effect specialized for workday

		Income		Pick time		Vacant run time		Order num	
		AVG	SD	AVG	SD	AVG	SD	AVG	SD
8:00–12:00	rand	37.09	45.19	741.5	570.06	10374	2970.0	4.85	3.13
	DQN	146.09	26.68	1721	438.29	3242.7	1330.6	11.6	2.15
16:00–20:00	rand	30.93	51.13	608.4	378.98	10875	3695.5	4.2	2.46
	DQN	127.03	18.48	1532	371.65	4573.1	1029.7	10.1	1.87
20:00–24:00	rand	12.91	44.30	528.1	307.78	11756	2751.2	3.25	1.89
	DQN	130.22	11.81	1503	368.30	4064.6	1101.5	9.9	1.51

Table 3. Training effect specialized for weekend

		Income		Pick time		Vacant run time		Order num	
		AVG	SD	AVG	SD	AVG	SD	AVG	SD
8:00–12:00	rand	39.22	42.43	756.9	544.40	10089	2871.8	5.05	2.89
	DQN	142.17	20.79	1463	423.8	3470.5	1344.7	9.9	2.30
16:00–20:00	rand	59.78	46.41	915.4	505.69	8643.4	2920.3	6.3	3.12
	DQN	136.33	26.23	1569	305.78	3845.7	1427.6	10.7	1.52
20:00–24:00	rand	34.55	52.83	602.0	478.06	10589	3532	4.15	3.10
	DQN	134.28	25.77	1336	372.94	4160.5	1530.2	9.1	1.81

At last we specifically compare the income in the three tables and list the bar chart. Wee see the income of situation 'all' is almost in the middle of other two situation 'workday' and 'weekend' in all three time periods. It brings small improvement by the workday and weekend specialized training, seeing the income listed in '20–24'. In our experiment, though the improvement between the 'DQN' and 'rand' is remarkable, the train specialized for the workday and weekend have rather less improvement (Fig. 5).

```
Algorithm 1 DQN for guiding taxis without clients
input: replay memory size $M$, episode times $N$, discount factor $\gamma$, time period for train.
output: Q-network Q(s,a)
 1: Initialize replay memory M, evaluation-network $Q$ and target-network $\hat{Q}$;
 2: for episode $i = 1, ..., N$ do
 3:     Initialize the taxi at random location and needed time;
 4:     take an action according to the state and the greedy policy;
 5:     run as the action and record the reward;
 6:     while order exists do
 7:         run the taxi to pick the passenger and record the reward;
 8:         run as the order and record the reward;
 9:     end while
10:     store transition$(s_i, a_i, r_i, s_{i+1})$ into the buffer;
11:     Randomly sample batch size of transitions$(s_j, a_j, r_j, s_{j+1})$;
12:     let input be $s_j$, target be $r_j + \gamma \max_a \hat{Q}(s_{j+1}, a)$;
13:     update the parameters of Q;
14:     Every C steps reset $\hat{Q} = Q$;
15: end for
```

Fig. 5. The picture contains the algorithm 1.

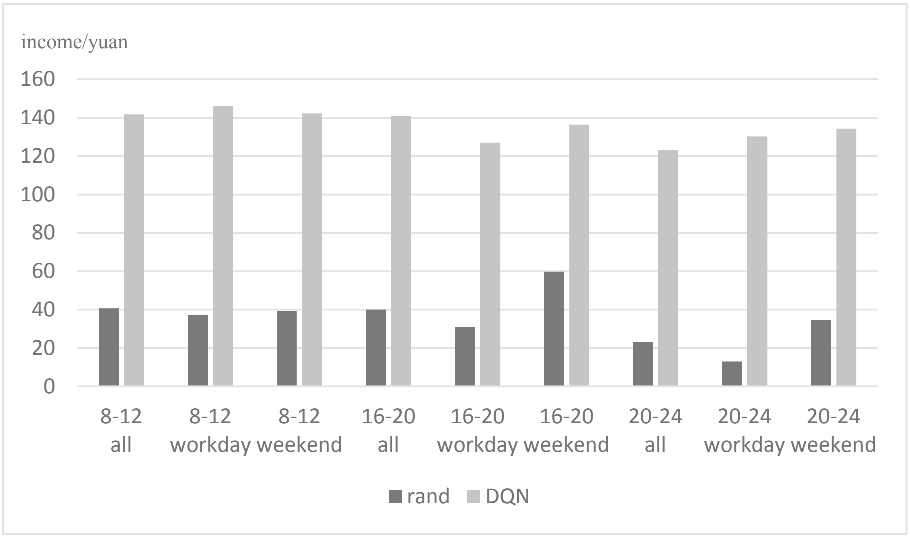

Fig. 6. The picutre shows the income in three periods of time respectively in three training situations. For example, ‘8–12’ is a simplified expression which means the period is from 8:00 to 12:00. And ‘all’, ‘workday’, ‘weekend’ stands for the situation in Tables 1, 2, and 3.

5 Conclusions and Future Work

This paper provides a designed model for the taxi cruising strategy according to an order data set and a method for training the cruising strategy by using DQN strategy. The taxi running model is adjustable to the desired precision and computation consumption by changing the parameter in the model. Then we compare the trained cruising strategy with a random strategy and show the significant improvement.

In our paper, we use the possibility of the orders taken by other taxis to simulate the existence of other taxis, which may cause some degree of inconsistency with the real world about whether there can be an order. And since we only have one taxi in the simulation world, we cannot test the competition situation when many taxis run in a typical trained strategy. It should be considered in the future work about a simulation and new model of multiple taxis.

References

1. Wang, H.: A Practice of Smart City-Intelligent Taxi Dispatch Services with Real-Time Traffic and Customer Information, 1st edn. Beihang University Press, Beijing (2011)
2. Dial, B.: Autonomous dial-a-ride transit introductory overview. Transp. Res. Part C Emerg. Technol. **3**(5), 261–275 (1995)
3. Zeng, W., Wu, M., Sun, W., Xie, S.: Comprehensive review of autonomous taxi dispatching systems. Comput. Sci. **47**(05), 181–189 (2020)
4. Powell, W., Huang, Y., Bastani, F., Ji, M.: Towards reducing taxicab cruising time using spatio-temporal profitability maps. In: Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases (SSTD 2011), pp. 242–260. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22922-0
5. Liu, L., Andris, C., Ratti, C.: Uncovering cabdrivers' behavior patterns from their digital traces. Comput. Environ. Urban Syst. **34**(6), 541–548 (2010)
6. Li, B., et al.: Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In: Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 63–68. (2011)
7. Zhang, K., Chen, Y., Nie, Y.M.: Hunting image: taxi search strategy recognition using sparse subspace clustering. Transp. Res. Part C Emerg. Technol. **109**, 250–266 (2019)
8. Maciejewski, M., Bischoff, J., Nagel, K.: An assignment-based approach to efficient real-time city-scale taxi dispatching. IEEE Intell. Syst. **31**(1), 68–77 (2016)
9. Nourinejad, M., Ramezani, M.: Developing a large-scale taxi dispatching system for urban networks. In: Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 441–446. IEEE (2016)
10. Wang, Y., Liang, B., Zheng, W.: The development of a smart taxicab scheduling system: a multi-source fusion perspective. In: Proceedings of the IEEE 16th International Conference on Data Mining (ICDM), pp. 1275–1280. IEEE (2016)
11. Liu, Z., Miwa, T., Zeng, W.: Shared autonomous taxi system and utilization of collected travel-time information. J. Adv. Transport. **2018**, 1–13 (2018)
12. Shen, Y., Chen, C., Zhou, Q.: Taxi resource allocation optimization under improved particle swarm optimization algorithm based on multi-chaotic strategy. J. Heilongjiang Univ. Technol. (Comprehens. Ed.) **18**(05), 72–76 (2018)
13. Xie, R., Pan, W., Shibasaki, R.: Intelligent taxi dispatching based on artificial fish swarm algorithm. Syst. Eng. Theory Pract. **37**(11), 2938–2947 (2017)
14. Verma, T., Varakantham, P., Kraus, S., Lau, H.C.: Augmenting decisions of taxi drivers through reinforcement learning for improving revenues. In: Proceedings of the 27th International Conference on Automated Planning and Scheduling, Pittsburgh, pp. 409–417 (2017)

15. Rong, H., Zhou, X., Yang, C., Shafiq, Z., Liu, A.: The rich and the poor: a Markov decision process approach to optimizing taxi driver revenue efficiency. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 2329–2334. Association for Computing Machinery, New York (2016)
16. Miyoung, H., Pierre, S., Stéphane, B., Wu, H.: Routing an autonomous taxi with reinforcement learning. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 2421–2424. Association for Computing Machinery, New York (2016)
17. Shi, D., Ding, J., Sai, E.: Deep Q-network-based route scheduling for TNC vehicles with passengers location differential privacy. IEEE Internet Things J. **6**(5), 7681–7692 (2019)
18. Wang, Y., Liu, H., Zheng, W.: Multi-objective workflow scheduling with deep-Q-network-based multi-agent reinforcement learning. IEEE Access **7**, 39974–39982 (2019)
19. Tseng, H., Luo, Y., Cui, S.: Deep reinforcement learning for automated radiation adaptation in lung cancer. Med. Phys. **44**(12), 6690–6705 (2017)
20. Carta, S., Ferreira, A., Podda, S.: Multi-DQN: an ensemble of deep Q-learning agents for stock market forecasting. Exp. Syst. Appl. **164**, 113820 (2021)
21. Sutton, S., Barto, G.: Reinforcement Learning: An Introduction. 2nd Edn. MIT Press, Cambridge (2018).
22. Watkins, H., Dayan, P.: Q-learning. Mach. Learn. **8**, 279–292 (1992)
23. Leemon, B.: Residual algorithms: reinforcement learning with function approximation. In: Proceedings of the International Conference on Machine Learning (ICML 1995), pp. 30–37. Elsevier, Amsterdam (1995)
24. Justin, B., Andrew, M.: Generalization in reinforcement learning: safely approximating the value function. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) Advances in Neural Information Processing Systems 7 (NIPS-94), pp. 369–376. MIT Press, Cambridge (1995)
25. Volodymyr, M., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)