Neural Computing and Applications (2022) 34:16275–16289
https://doi.org/10.1007/s00521-022-07255-9

S.I : NCAA 2021

Deep reinforcement learning for urban multi-taxis cruising strategy

Weian Guo[1,3] · Zhenyao Hua[2] · Zecheng Kang[2] · Dongyang Li[2] · Lei Wang[2] · Qidi Wu[2] · Alexander Lerch[4]

Received: 5 November 2021 / Accepted: 29 March 2022 / Published online: 27 April 2022

Abstract
Taxis play an important role in urban transportation system. Efficient taxi cruising strategies are helpful to alleviate urban traffic congestions, reduce pollution emission, attenuate greenhouse gas, and also provide a fast service for passengers. However, in real scenarios, taxis cruising strategies are mostly based on their own experiences. At unfamiliar urban areas or during off-peak hours, drivers usually have no good idea for an optimal cruising strategy, which causes a low efficiency for service and also increases taxi operation costs. Considering that it is difficult to construct an analytical model for the taxis scheduling and cruising, in this paper, we put forward a data-driven model for multi-taxis cruising based on reinforcement learning. Furthermore, an evolutionary reinforcement learning method is proposed, which aims at improving the exploration of reinforcement learning and enhancing reinforcement learning to maximize the global reward in multi-agent tasks. In the experimental part, two other kinds of deep Q-learning methods and a roaming strategy are employed in the comparisons. The results demonstrate the superiority of our proposed algorithm.

Keywords Urban transportation · Multi-taxis cruising · Data-driven model · deep Q-learning network

1 Introduction

As an important role in urban transportation system, taxis offer rapid, convenient and comfortable services for citizens' traveling. In many metropolises of China, both the number of taxi-cabs and taxi-passengers account from 10 to 20% in the whole city transportation system. Especially for certain critical areas, the proportion rises to 50 to 60% [1]. Therefore, the taxis service quality is crucial to urban transportation efficiency. Due to the huge number of taxis in urban transportation system, they may significantly contribute to transportation congestions, pollution and CO_2 emissions, waste of energies and so forth. To reduce the negative influence, taxi cruising strategy plays an important role. Currently, the cruising strategy for each taxi-cab

Zhenyao Hua, Zecheng Kang, Dongyang Li, Lei Wang, Qidi Wu and Alexander Lerch have contributed equally to this work.

✉ Dongyang Li
1710333@tongji.edu.cn

Weian Guo
guoweian@tongji.edu.cn

Zhenyao Hua
2032976@tongji.edu.cn

Zecheng Kang
2130806@tongji.edu.cn

Lei Wang
wanglei@tongji.edu.cn

Qidi Wu
qidi@tongji.edu.cn

Alexander Lerch
alexander.lerch@gatech.edu

[1] Sino-German College of Applied Sciences, Tongji University, Caoan Road, Shanghai 201804, Shanghai, China

[2] Department of Electronics and Information Engineering, Tongji University, Caoan Road, Shanghai 201804, Shanghai, China

[3] Shanghai Institute of Intelligent Science and Technology, Tongji University, Caoan Road, Shanghai 201804, Shanghai, China

[4] Center for Music Technology, Georgia Institute of Technology, North Avenue, Atlanta 30332, Georgia, USA

mostly depends on divers' own experiences. At unfamiliar urban areas or during off-peak hours, taxis drivers may have no good strategy for cruising. With the development of information technology, it is possible to apply artificial intelligence in taxis cruising to enhance the efficiency of taxis services. In particular, in the future development of robo-taxis, an artificial intelligence strategy is much useful in transportation network companies operation.

To date, there exist many researches focusing on addressing urban transportation issues. Public transportation scheduling problems [2] mainly include route design, timetable design and vehicle dispatching. Jiang et al. propose a two-level planning model, which takes timetabling and vehicle scheduling into consideration [3]. Bie et al. propose a scheduling method targeting against the bus routes overlapping problem, which is able to reduce passengers' traveling-time and the operation complexity of bus enterprises [4]. Except for the researches on single vehicle problems, multi-vehicles scheduling problem (MVSP) has also been widely investigated, such as the variant capabilities coordination aiming at addressing given transportation tasks [5, 6] and the multi-vehicle scheduling methods based on the dynamic programming [7, 8].

In particular, as an important part in transportation system, taxis also draw many attentions. Various conditions and factors are involved in the researches in solving the taxi scheduling problems. Focusing on the driving style of the driver, Chen et al. propose a multi-task learning framework of semi-Traj2Graph, which uses a graph neural network to judge the driving trajectory of GPS to achieve the best accuracy [9]. Taking the advance matching, time-varying price and other traffic service conditions into account, Guo et al. propose a force-oriented approach, which focuses on improving passengers' benefits under time-varying price [10]. Lee et al. put forward a method based on the real-time traffic conditions to ensure the passengers' demands and reduce the time that taxis cost to pick up passengers [11]. In their experiment, simulations with respect to the Singapore transportation show that the picking up time can be reduced by 50%. Furthermore, [12–15] design various taxis distribution models to evaluate the saturation degree of different regions, based on which different taxis dispatching methods are proposed. For instance, Liu et al. provide a scheduling model based on clients demands, taxis distribution and routing situation with respect to both the real-time data and historical data [15]. In [16–25], the authors focus on building the traveling models based on taxis' trajectories including drivers' actions and passengers' demands. To name a few, Bai et al. employ Nash equilibrium theory to improve the taxi service efficiency [19]. Chen et al. propose a two-stage framework, which is able to extract historical taxi trajectories from the existing taxi and POI resources and optimize the distribution and routing of packages by employing Adaplan (An adaptive taxi scheduling algorithm)[20]. Seow et al. present a distributed dispatching architecture, where taxis negotiate and coordinate with each other to ensure passengers' demands [21]. Dial-a-Ride problem (DARP) is also a well-defined problem about assigning taxis to specified transportation requests [22]. Attanasio et al. investigate a dynamic DARP where there is no priori knowledge of the passengers' requests [23], they try to respond passengers' requests as much as possible with operational constraints. Furthermore, evolutionary algorithms are also widely employed for such kinds of problems [24, 25].

For taxis operation, the cruising time directly influences service efficiency. When a taxi is non-hailing, drivers usually cruise based on roaming strategy or their own experience. Once the strategy is not suitable for the current time-periods or areas, it will cause an increment of non-hailing time, more expenses and wasted energies [26]. To address such issues, many studies have been conducted to design cruising strategy. Swarm intelligence algorithms, including particle swarm optimization and fish swarm algorithm et al., are widely used to realize a more sensible global taxi-cabs resources allocation [27, 28]. Machine learning-based strategies are also utilized to predict passengers' demands to help allocating taxi resources. Xu et al. employ recurrent neural networks to analyze the distribution of taxis trips and passengers demand [29]. In addition, Markov decision process [30, 31] and Q-learning [32, 33] are adopted to conduct taxis dispatching by considering drivers revenue efficiency and passengers' location. Multi-agent reinforcement learning has been used to scheduling taxis to meet passengers' demands [34]. For instance, Liu et al. propose a META framework based on multi-agent reinforcement learning to solve the problem of taxi supply and demand [35].

However, most of the current researches focus on the macroscopic view, but lacks of related researches from taxi agents standpoint for cruising strategy. On the other hand, thanks to the appearance of smart phones and GPS technology, it is currently very popular to book and call taxis service online, such as the service provided by Uber, Didi and Baidu et al. With information platforms, the locations of both taxis and passengers can be shared for calling and responding taxis services [36]. Therefore, in this paper, we employ Q-learning to construct a data driven model to

analyze passengers' demand and guide taxi drivers for cruising to reduce non-hailing time and increase drivers' income profit. Furthermore, to enhance the exploration of Q-learning methods and improve Q-learning for handling multi-agent task, an evolutionary reinforcement learning method is proposed. The main contributions of this paper are listed as follows.

- A Q-learning-based taxi cruising strategy is proposed, where the taxi cruising problem is formulated as a reinforcement learning case with respect to maximizing the taxis' profit.
- An evolutionary reinforcement learning method is proposed, where a mutation-based Q-network evolutionary strategy and a dual-agent iterative exploration are introduced for improving the reinforcement learning for exploration and multi-agent tasks.

The rest of this paper is organized as follows. In Sect. 2, we construct models for taxi cruising based on the view of drivers' benefits. Section 3 describes the proposed Q-learning-based taxi cruising strategy and the proposed evolutionary reinforcement learning algorithm. Section 4 conducts the experiments and simulations to evaluate the proposed model and method. Finally, We end this paper in Sect. 5 and present our future work.

2 Problem description

2.1 Modeling

In general, taxi cruising strategy is based on drivers' own experience, for different kinds of purposes, such as total profit or total accounts of service or special destination, et al. Besides, some other factors in the real scenarios are also supposed to be considered, which involves the taxis' positions and drivers' working time. There also exists uncertainties in passengers demands, because in different periods of one day or in different dates, the transportation conditions are not similar. Besides, the weather conditions and passengers preferences will also affect the transportation conditions so that it is difficult to construct an analytical formula for taxis dispatching system just by human-beings experience. 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' income profit.

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

where P is the drivers' total profit, G_i is the income for the ith passengers' request, while F_j is the cost of energy consumption in the jth driving which involves three statuses, namely carrying status, picking status and cruising status. Once a driver responds a passenger's order, the taxi stops cruising status and go into picking up status to arrive at the starting location of the passenger's order. After that passengers are taken by the taxi, the taximeter begins to count and the driver will be in carrying status. The taxi will be in cruising status until the taxi finishes the current order.

A typical pattern of statuses transition is shown in Fig. 1. In the figure, the x-axis is the time line, while the y-axis represents the three statuses. The taxi starts with the cruising status during the period of $[t_0, t_1]$. When a taxi Tax_c is cruising, it will search for orders generated within two minutes in current grid. First, determine whether there exists other taxis in the grid, then 1. If there are other taxis, Tax_c will be randomly assigned an order or not. If Tax_c receives an order, its state will be switched to the picking state, and Tax_c will continue to maintain the cruise state if it is not assigned to the order. 2. If there exists no other taxi in the grid, Tax_c will be directly assigned an order, and the status will be switched from cruising to picking. When it responds to an order at t_1, its status changes to picking status and meanwhile the taxi moves to the starting location of the order. After arriving at the starting location t_2, the taxi picks up the passengers. Then the taxi's status changes to carrying status. Finally, when the taxi finishes an order task at t_3, the status changes back to the cruising status. In addition, taxis will also encounter the period of $[t_5, t_8]$. In this period, after finishing one order at t_5, the taxi immediately responds to another order and its status skips the cruising status but directly changes to picking status. In the bottom of Fig. 1, we list the periods which are related to G_i and F_j. A taxi has income G_i only at carrying status, while pay energy costs F_j at all three statuses. According to Fig. 1, to enhance the profit P in (1), decreasing the cruising time is a main approach, and therefore service efficiency is promoted.

To construct a learning environment for taxi agents, a whole city is divided into $N * N$ grids. When a taxi transits to the cruising status, it uses the grid index as the function input to determine which grid it should go to in the next step. The strategy is defined as

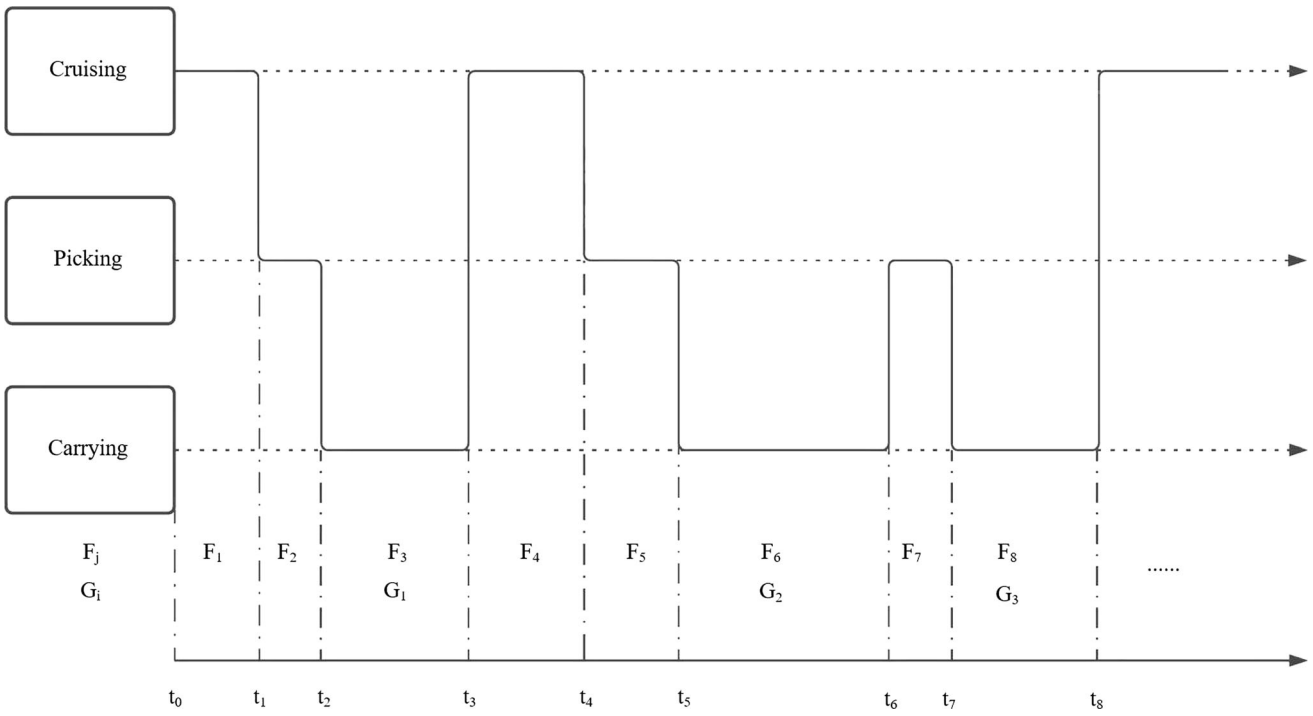


Fig. 1 An example illustrating the possible statuses transition of a taxi as time passes

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

where S is the strategy function, T is the target grid represented by grid index, (x, y) is the current location of a taxi. We do not design path planning in this paper, but employ GaoDe Map API to calculate traveling time and income for each order.

Besides the above descriptions in the modeling, some preliminaries and assumptions are also made as follows.

- For a booking order, it includes start position, destination position, and the time of launching a booking.
- Drivers only know the start position before picking up the passengers. After picking up, drivers will know the destination position and cannot refuse the order.
- For a grid which contains more than one taxi, they have competition relationship. Only one taxi can respond an order with a predefined probability and passengers cannot choose taxis.
- Taxis only serve one order at a time and will not respond to any other order during its service period.
- Due to limitation of fuel and the service policy, a taxi cannot continuously work for a long time. Therefore we split one day into several segments.

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 work flow scheduling [37], automated radiation adaption in lung cancer [38], stock market forecasting [39], et al. 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 [40]. At each time step t, an agent is described by a state s_t from a state space, while it will also select a feasible action a_t from the action space, following a policy $\pi(a_t \mid s_t)$. Then the agent receives 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 $\pi(a_t \mid s_t)$ to maximize the accumulated rewards with a discount factor $\gamma \in [0, 1]$. The accumulated reward is defined as

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

where R_t is the accumulated rewards at the tth time step, γ is the discount factor and r_{t+k} is the expected reward got at the $(t + k)$th time step. In this paper, Q-network learning

[41] is employed to optimize the policy π. In Q-network, Q is used to depict action value and defined as

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

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

$$\pi(s) = \arg\max_a(Q_\pi(s,a)), \tag{5}$$

where $\pi(s)$ is the policy function to get the selected action a by solving a maximization function. Namely the action a will help an agent get the largest expected accumulated reward based on 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, and fully incremental. The TD error is given as

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

and the update rule is set as

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

where η is the learning rate. Q-learning converges to the optimum action-values with probability of 1 so long as all actions are repeatedly sampled in all states and the action-values are represented discretely [42]. Considering that there are thousands of grids in a city and the scale of (s, a) pairs will be too large, it is not feasible to build a Q-table. Here we use a deep neural network to represent the Q function. The detail of the network design is introduced in Sect. 3. For the network, the input of neural network is the agents' state, while the output is the Q-value of every action. The parameters in the neural network are updated through gradient descent as the update rule in (7). The Q function may not stay stable when the function is in the form of approximation [41, 43]. Therefore, to stabilize the learning process, there are still many essential operations such as replacing buffer, target network, hyper-parameter tuning and so forth [44]. To balance the exploration and exploitation, epsilon greedy is used in this paper. Replay buffer is a first-in-first-out queue to save the experience transition (s,a,r,s') so that the experience can be used repeatedly in an off-policy way. During training a batch of experience transitions will be uniformly sampled from the experience buffer to optimize the policy network. The interaction with the environment can be more efficient and the training can be more stable. For the target network, it is used to compute the target value of ta station-action Q-value, exactly as the first term in (6) where the Q is the target network. It is updated to the policy network every few episodes, so that the target value of the policy neural network can be more stable and it can also follow the changing policy network.

3 Policy optimization

3.1 Policy design

In this paper, we employ Q-network as the basis of the taxi cruising strategy. To be specific, a deep neural network is adopted to estimate the Q-value of an action at each state for a taxi. As shown in Fig. 2, on the left part, city grids are depicted which are used to record the current location of a taxi, namely the starting point for one cruising. For each starting point, it has $N * N$ candidate target grids. Correspondingly, it means that a taxi-cab has $N * N$ candidate actions. The outputs of the Q-network are shown in the right part in Fig. 2. To pursue a maximal profit, namely the reward, a taxi agent considers the energy consumption and the potential income. Therefore, r in (3) is set as the P in (1), and Q-value is expressed and updated by (4), (6), (7). For different time periods, the strategy may vary due to the fluctuation of passengers' demands. Therefore, a whole day is divided into several periods in training. Afterwards, the next action for each taxi in a specific state is decided by a greedy strategy.

3.2 Training program

The agents (taxis) interact with the environment, which is explained in Sect. 2.1, and each experience transition tuple (s,a,r,s') is put into a replay buffer. Then a batch of transitions are sampled from the replay buffer to train the policy network by a Double DQN (DDQN), which is proposed in [45] to alleviate the over-estimation of Q-values in common DQN. Since there is only one policy network, only one target network is set to help produce the target Q-value and its parameters are updated to the latest policy network for every C episodes.

3.3 Evolutionary Q-learning strategy

Despite of the success of various RL training methods[46], reinforcement learning methods usually fail on exploration in both the state and parameter space: the former focuses on generating diverse training data while the later aims to improve the global optimization ability in training [47]. The main reasons are that, first, a single agent is usually less effective in generating diverse training data [48]; second, the optimal hyperparameters for specific exploration operators are difficult to determine [49]; third,

gradient-based methods in general tend to be less effective in exploration.

In addition, for multi-agent tasks, the reinforcement learning methods are potentially less effective in ensuring the global learning ability, especially when only one Q-learning network is used. The reason is that all the agents are likely to take the actions with high value, resulting in adverse impacts on taxis cruising.

To solve the above issues, an evolutionary Q-learning strategy is proposed as following.

3.3.1 Evolutionary network optimization strategy

Since evolutionary algorithms (EAs) are found less effective in large scale optimization, we propose to optimize the initialization parameters of the Q-network instead of optimizing the weights and biases of the network. After that, new agents can be obtained based on the optimized initialization parameters settings. Here, the initialization parameters represent the parameters of the distribution for sampling the weights and biases. By this means, the proposed method is able to reduce the number of decision variables optimized by EAs and adopt multiple agents to enhance the exploration to avoid a single agent being trapped in local optima. The proposed evolutionary network optimization strategy (ENOS) is described as follows.

First, a set of parameter initialization settings (The candidate solutions of EAs) are randomly generated and will be used to generate new Q-network in the next step. More specifically, four parameter initialization settings can be found in the ith hidden layer with respect to a fully connected layer: $p_{wi,1}$, $p_{wi,2}$, $p_{bi,1}$ and $p_{bi,2}$. Here, $p_{wi,j}$ and $p_{bi,j}$ with $j = [1, 2]$ represent the the characteristics (i.e., the maximum and minimum values of an uniform distribution) of the distribution for sampling weights and biases, respectively.

Second, a set of Q-network will be generated with these parameter initialization settings. Since the stability of the agent optimization might be affected by huge policy changes [50], the network generator (NG) is based on a random mutation strategy shown in Algorithm 1. The new network are constructed on the basis of the current best network with respect to the evolved parameter initialization settings. This design not only diversifies the generated Q-networks for exploration, but also ensures that the generated Q-networks inherit the main settings of the currently best agent.

Third, gradient-based methods are executed on the generated networks for exploitation. Since the evolutionary operation potentially leads to invalid (*inf*) values in weights and biases after back propagation due to dramatic changes caused by mutation or unexpected training data, this paper defines following feasibility detection to ensure stable optimization: (1) for the best network at each epoch, the policy will remain unchanged if any invalid weights or biases are detected after executing the policy gradient, and (2) for the rest of the networks at each epoch, the invalid weights and biases after executing the policy gradient will be replaced with 0. Finally, each Q-network interacts with the environment and the corresponding reward is adopted as its fitness value as well as the fitness value of the corresponding parameter initialization setting. After that, the best Q-network is selected and the set of parameter initialization settings will be evolved by EAs to improve their quality and used to generate new Q-network for exploration at the next epoch. The pseudo-codes of ENOS are shown in Algorithm 2.

Algorithm 1 The pseudo-code of NG.

Require: The current best Q-network $\mathcal{A}^*$, a specific parameter initialization setting $\mathcal{I}$, mutation rate $\mathcal{R}$, the number of hidden layers $\mathcal{N}_{\mathcal{H}}$

Ensure: An offspring network $\mathcal{A}^*_{\wr}$ of $\mathcal{A}^*$

$\vec{\mathcal{P}}^* \leftarrow$ Collect the parameters of $\mathcal{A}^*$;
$\vec{\mathcal{P}} \leftarrow$ Parameter initialization based on $\mathcal{I}$;
$Pos \leftarrow \emptyset$;
for $i = 1$ to $\vec{\mathcal{P}}$ **do**
    if $rand < \mathcal{R}$ **then** $Pos \leftarrow Pos \cup i$;
    end if
end for
$\vec{\mathcal{P}}^*[Pos] = \vec{\mathcal{P}}[Pos]$;
Produce $\mathcal{A}^*_{\wr}$ with $\vec{\mathcal{P}}^*$;

Algorithm 2 The pseudo-code of EANS.

Require: The current best agent $\mathcal{A}^*$, the set of parameter initialization settings $\mathcal{S}_\mathcal{I}$, training data D

Ensure: The best Q-network $\mathcal{A}^*$ and the evolved parameter initialization settings $\mathcal{S}_\mathcal{I}$ after one optimization step, the fitness values of different parameter initialization settings Fit

$\mathcal{A}^* \leftarrow$ Execute gradient-based optimization with D;
Evaluate $\mathcal{A}^*$;
$Fit \leftarrow \emptyset$;
$k = 1$;
for $\mathcal{I}$ in $\mathcal{S}_\mathcal{I}$ **do**
    $\mathcal{A} \leftarrow$ Algorithm. 1;
    $\mathcal{A} \leftarrow$ Execute gradient-based optimization with D;
    $Fit[k] \leftarrow$ Evaluate $\mathcal{A}$;
    $\mathcal{A}^* \leftarrow \mathcal{A}$ If $\mathcal{A}$ is better than $\mathcal{A}^*$;
    $k = k + 1$;
end for
Evolve $\mathcal{S}_\mathcal{I}$ by EAs with respect to Fit

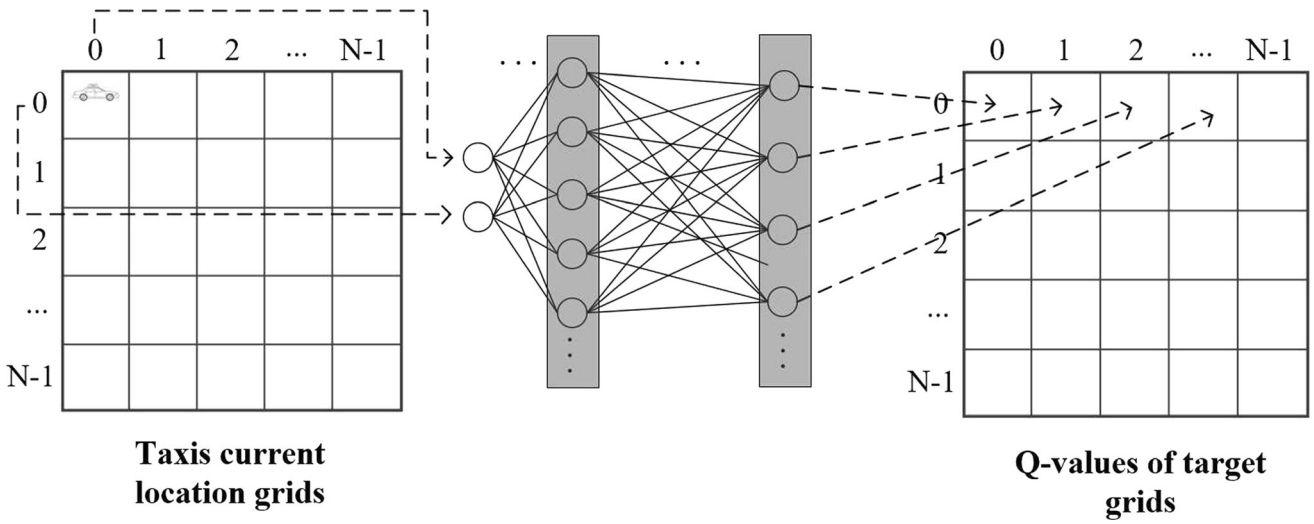


Fig. 2 The construction and description of Q-network

3.3.2 Dual-agent iteratively exploration strategy

Producing training data for reinforcement learning with multiple agents has been found effective in improving the exploration in state space [51]. However, excessively diversified training data may lead to adverse impacts to agents' training [52]. On the other hand, it is difficult to define how good an agent (Q-network) should be in order to serve the training data generation, since the training data generated by poor agents may be less valuable. To this end, a dual-agent iterative exploration strategy (DAES) is proposed as follows.

First, in odd-numbered episodes, the best two Q-networks generated in Algorithm 2 will interact with the environment to generate training data while in even-numbered episodes, only the best Q-network is adopted to produce the training data. Compared with the reinforcement learning methods with single agent, using two Q-networks allows generating more diverse training data. On the other hand, limiting the number of Q-networks to two potentially prevents the training data from being excessively diversified.

Second, to reduce the computational cost, all Q-networks generated by EAs except for the currently best two will be eliminated.

Together, the pseudo-codes of DAES are shown in Algorithm 3.

Algorithm 3 The pseudo-code of DAES.

Require: Q-network set $\mathcal{S}_{\mathcal{A}}$; the current number of episode n_e
Ensure: The best two Q-networks set $\mathcal{S}'_{\mathcal{A}}$ and the training data D;
$Fit \leftarrow$ evaluate each Q-network in $\mathcal{S}_{\mathcal{A}}$ based on the interaction between the Q-network and environment;
$S^o_A \leftarrow$ select the best two Q-networks from $\mathcal{S}_{\mathcal{A}}$;
if n_e is an odd number **then**
    $D \leftarrow$ collect the training data based the interaction between the best two Q-networks and environment;
else
    $D \leftarrow$ collect the training data based the interaction between the best Q-network and environment;
end if

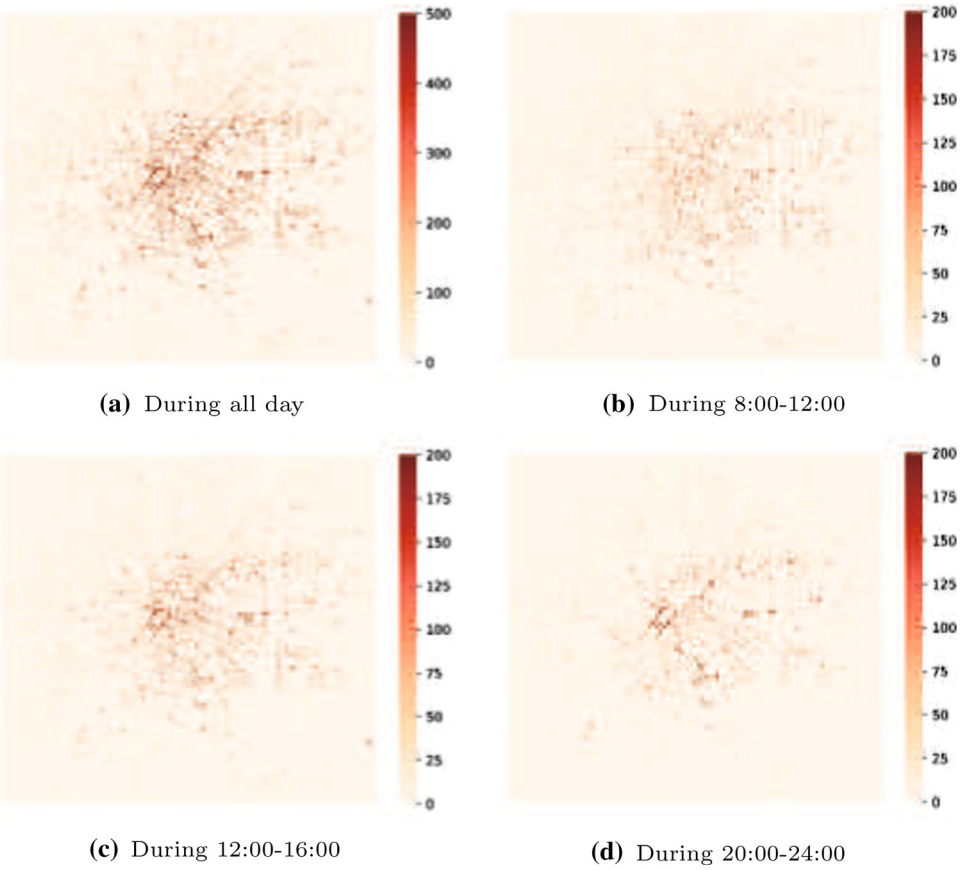
3.3.3 Proposed evolutionary network optimization strategy

By combining the proposed ENOS and DAES, the evolutionary network optimization strategy is proposed, which is named with ERL in this paper for simplicity. The corresponding pseudo-code of ERL are shown in Algorithm 4.

Algorithm 4 The pseudo-code of ERL.

Require: The size of agent set $Npop$, the mutation rate $\mathcal{R}$, the structure definition of agent $\mathcal{A}_{\mathcal{S}}$, the terminal criterion $\mathcal{T}$
Ensure: The best agent $\mathcal{A}^*$
Generate a set of agent $\mathcal{S}_{\mathcal{A}}$ with respect $Npop$ and $\mathcal{A}_{\mathcal{S}}$;
Generate a set of parameter initialization settings $\mathcal{S}_{\mathcal{I}}$;
The best two agents set $\mathcal{S}_{\mathcal{A}}$ and the training data $D \leftarrow$ Algorithm. 3;
$Fit \leftarrow \emptyset$;
$epoch = 1$;
while $\mathcal{T}$ is not satisfied **do**
    $\mathcal{A}^* \leftarrow$ Select the current best model from $\mathcal{S}_{\mathcal{A}}$;
    Training the best two agents with specific gradient-based methods and D;
    Feasibility detection for the best two agents;
    Evaluate the best two agents;
    for $i = 1$ to $Npop$ **do**
        $\mathcal{A}^*_i \leftarrow$ Algorithm. 1 based on $\mathcal{A}^*$ and $\mathcal{S}_{\mathcal{I}}[i]$;
        Training $\mathcal{A}^*_i$ with specific gradient-based methods and D;
        Feasibility detection for $\mathcal{A}^*_i$;
        $\mathcal{S}_{\mathcal{A}}, \sim \leftarrow$ Algorithm. 3 with respect to $\mathcal{S}_{\mathcal{A}} \cup \mathcal{A}^*_i$;
    end for
    Apply EAs to evolve the parameter initialization settings $\mathcal{S}_{\mathcal{I}}$;
    $\mathcal{S}_{\mathcal{A}}, D \leftarrow$ Algorithm. 3;
    $epoch = epoch + 1$
end while

It can be found that the proposed method combines the advantages of EAs and the policy gradient approach and is more efficient with respect to computational resources: first, it improves the exploration in the parameter space, since it adopts EAs to evolve a set of Q-network. Second, it ensures an enhanced exploration in state space compared with the methods with single Q-network. The reason is that two Q-networks are used to produce the training data, leading to more diverse interactions between agents and environments. Third, the proposed method can adapt multi-



(a) During all day **(b)** During 8:00-12:00

(c) During 12:00-16:00 **(d)** During 20:00-24:00

Fig. 3 A heat map describing order start points distribution during different time periods in Chengdu

agent tasks, since the evolutionary process aims at selecting the Q-networks that can get the best global reward.

4 Experiments and results

4.1 Dataset

In this section, we conduct experiments to validate the performance of our proposed model and algorithm. The data are obtained from GAIA Open Dataset provided by DiDi company[1], which depicts taxis operations in Chengdu city, China. 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 TianFu-FuRong Garden to the east-north corner named Qilong Temple. Based on the information, we also calculate the payment price and time duration of each order by Ali-cloud. In this experiment, the continuous working time duration for a taxi is set as 4 hours: 8:00 to 12:00, from 16:00 to 20:00 and from 20:00 to 24:00, respectively.

The heat maps of start points during all days, duration of 8:00–12:00, duration of 16:00–20:00, duration of 20:00–24:00 are listed in Fig. 3. The corresponding heat maps of destination points of the four time-durations are presented in Fig. 4. From the Figs. 3 and 4, one can see that the distribution of the start and end points of the order. The center of the area is where the order density is greatest, and 12:00-16:00 is the time period with the largest number of orders, and 20:00-24:00 is the time period with the least.

4.2 Compared algorithm

To test the proposed model and ERL, two Q-learning algorithms are employed in the comparison: DTDE [53] and CTDE [54]. In DTDE, independent DQN is employed, the action for each taxi-cab is decided by a greedy strategy shown in (8).

$$a = argmax_a(Q_\pi(s,a)). \tag{8}$$

In CTDE, single DQN is utilized, the action for each taxi-cab is decided by a greedy strategy shown in (9),

$$p(a_i \mid s) = \frac{e^{Q(s,a_i)/T}}{\sum_k^K e^{Q(s',a_k)/T}}, \tag{9}$$

[1] https://outreach.didichuxing.com/research/opendata/.

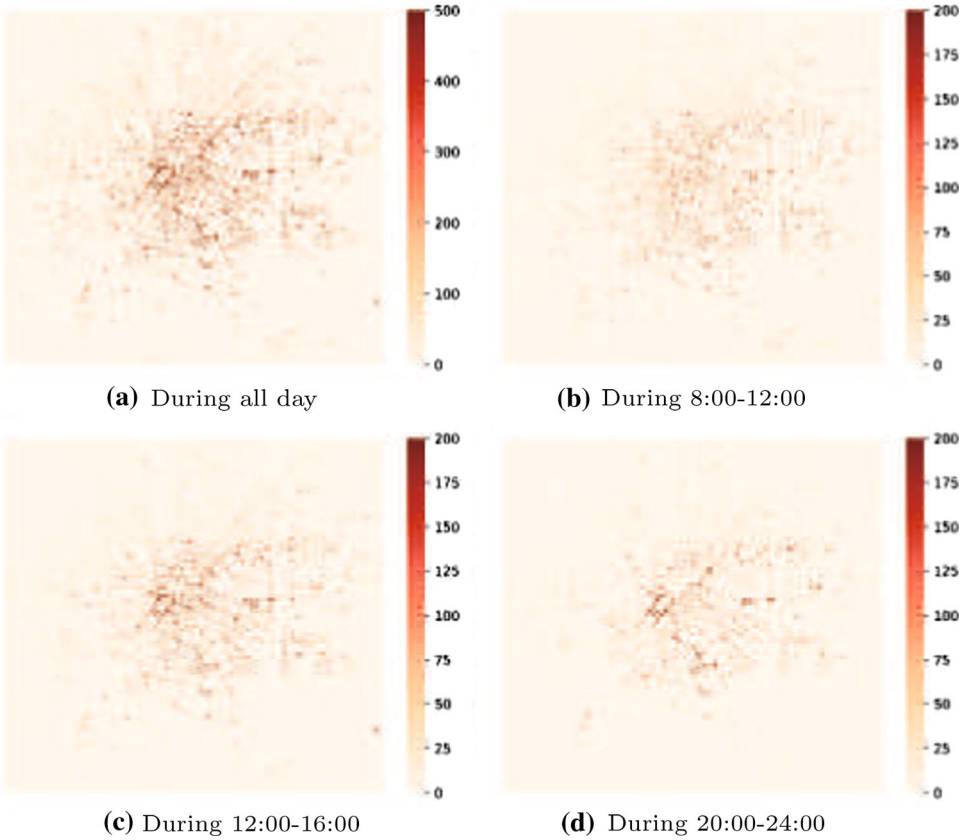


Fig. 4 A heat map describing order end points distribution during different time periods in Chengdu

Table 1 Test records based on synthetic data set

		Profit		Pick time		Vacant run time		Order num	
		AVG	SD	AVG	SD	AVG	SD	AVG	SD
8:00–12:00	Roaming	31.66	2.18	689.73	15.39	10237.72	122.29	3.54	0.081
	CTDE	66.11	2.76	1074.65	19.07	7414.84	122.59	5.583	0.11
	DTDE	65.11	1.83	952.88	12.99	8007.70	85.84	4.98	0.05
	ERL	56.28	1.54	1022.42	17.27	8057.11	105.81	5.39	0.07
12:00–16:00	Roaming	33.74	1.37	698.93	10.26	10127.26	84.41	3.58	0.05
	CTDE	67.89	2.12	1075.81	17.20	7356.66	112.59	5.5713	0.08
	DTDE	66.55	2.01	931.40	14.92	7991.28	83.47	4.76	0.06
	ERL	62.48	1.67	1018.66	16.79	7888.56	74.92	5.31	0.08
20:00–24:00	Roaming	32.49	1.07	683.00	11.28	10242.27	115.02	3.52	0.06
	CTDE	66.25	2.88	1096.48	30.32	7398.04	111.86	5.63	0.14
	DTDE	64.79	2.30	1031.18	19.20	7733.98	77.52	5.34	0.08
	ERL	58.10	1.63	1062.97	12.59	7773.80	88.67	5.55	0.06

where T is a temperature parameter in Boltzman distribution, which is used to control action randomness, $Q(s, a_i)$ is calculated based on (4), e is exponent arithmetic operator, k is the total number of possible actions. In (9), we obtain the probability of each action (a_i) occurrence at state s.

Furthermore, roaming scheme (taxis cruise randomly) is also investigated in the experiment, where each taxi randomly selects the target grid selected in the whole city.

Table 2 Test records based on workday data set

		Profit		Pick time		Vacant run time		Order num	
		AVG	SD	AVG	SD	AVG	SD	AVG	SD
8:00–12:00	Roaming	31.64	1.777	697.05	11.33	10217.96	110.28	3.59	0.061
	CTDE	− 18.97	2.81	220.95	30.62	12996.18	179.43	1.12	0.14
	DTDE	64.83	1.40	957.60	9.06	7983.28	75.43	5.00	0.04
	ERL	57.01	1.38	1017.75	9.30	8121.21	74.36	5.37	0.03
12:00–16:00	Roaming	33.42	1.39	693.08	11.87	10162.71	88.70	3.54	0.05
	CTDE	50.12	1.76	830.55	12.69	8812.56	97.41	4.24	0.06
	DTDE	67.20	1.73	943.69	7.52	7901.90	60.78	4.87	0.03
	ERL	57.36	1.54	1026.57	11.42	7999.85	70.74	5.40	0.05
20:00–24:00	Roaming	21.70	2.35	585.40	22.01	11106.07	144.77	3.02	0.10
	CTDE	20.14	2.31	557.78	17.15	10589.62	128.64	2.92	0.09
	DTDE	63.87	1.6	1022.72	12.88	7769.53	60.60	5.29	0.06
	ERL	55.27	1.81	1099.47	17.11	7783.25	66.76	5.71	0.09

Table 3 Test records based on weekend data set

		Profit		Pick time		Vacant run time		Order num	
		AVG	SD	AVG	SD	AVG	SD	AVG	SD
8:00–12:00	Roaming	32.49	1.07	683.00	11.28	10242.26	115.02	3.52	
	CTDE	−35.01	0.73	52.05	5.56	13998.56	50.48	0.26	0.02
	DTDE	66.20	1.17	970.23	12.31	7903.56	79.65	5.04	0.05
	ERL	56.99	1.73	983.50	15.41	8119.78	130.41	5.24	0.08
12:00–16:00	Roaming	34.81	1.02	712.38	8.05	10012.78	136.69	3.63	0.03
	CTDE	−32.43	0.84	54.13	2.73	14064.28	40.33	0.30	0.02
	DTDE	67.54	1.31	922.40	12.76	8051.55	110.93	4.69	0.04
	ERL	61.13	1.70	1008.49	14.47	7796.08	96.63	5.40	
20:00–24:00	Roaming	24.20	1.26	602.06	9.90	10985.41	115.00	3.11575	0.05
	CTDE	−16.04	1.44	205.25	11.82	12963.31	104.73	1.12	0.06
	DTDE	64.02	1.48	1001.16	12.81	7877.39	91.6	5.18	0.047
	ERL	60.69	1.65	1103.90	13.74	7592.17	115.15	5.83	0.06

4.3 Experimental settings

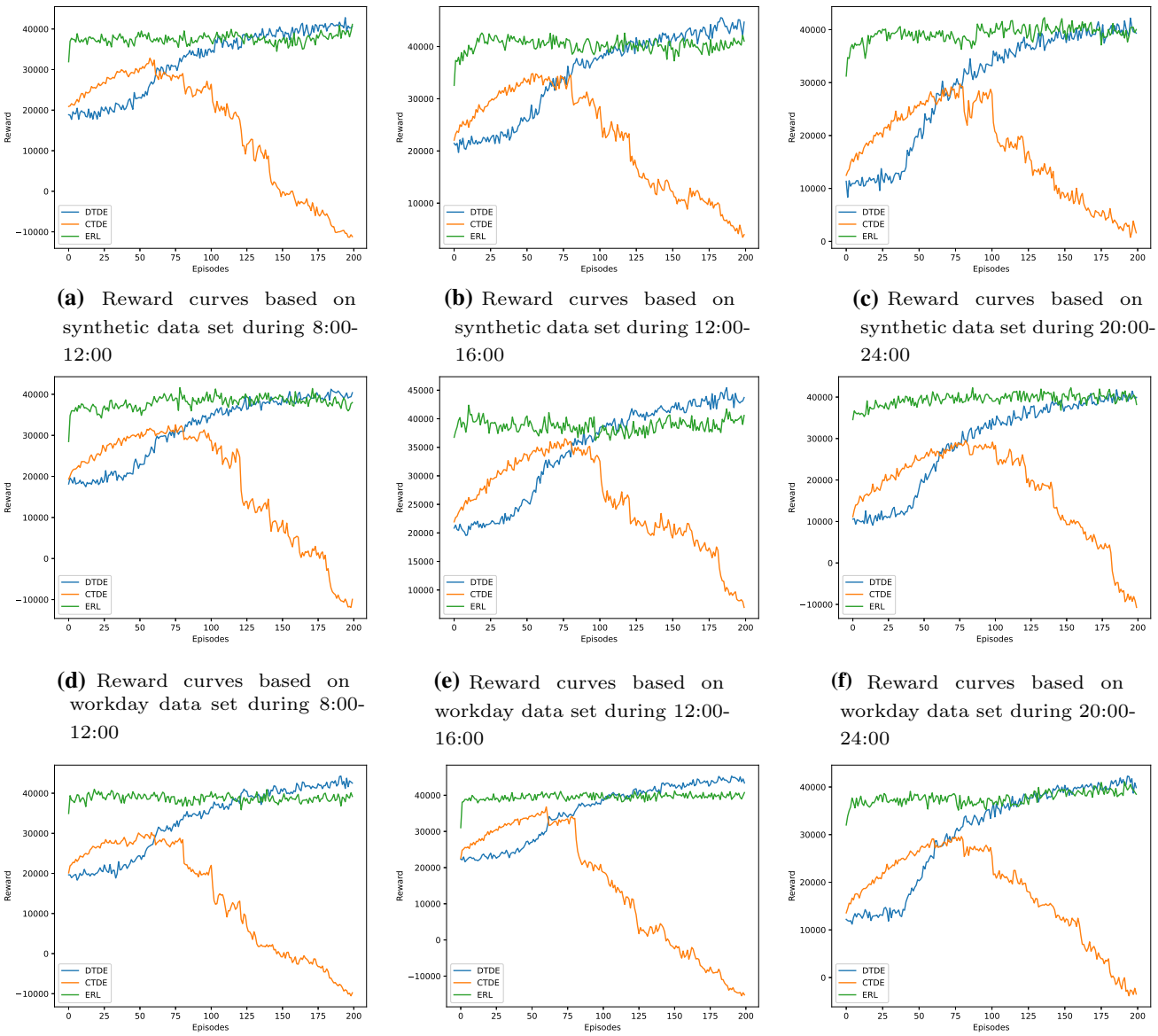
In the experiment, Chengdu city is divided into grids with size of 10. The Q-network is a mapping from a taxi-cab state to the Q value of every action, so the number of input nodes is 2, while the number of the output nodes is 100. For the hidden layers, the layers number is set as 3, and for the three layers, the nodes are set as 20, 80, 160, respectively. The taxi number is set to 1000. The batch size of DTDE and CTDE is set to 256, and the Q-network will be updated with TD algorithm, the replay memory size is set to 6000.

In the proposed evolutionary reinforcement learning method, referred as ERL in following for simplicity, CTDE is selected as the base model to conduct policy gradient and APSODEE [55] is adopted in the evolutionary process. The batch size is set to 512 and the replay memory size is set to 5120. The mutation rate in the network generator is set to 0.001, the population size, ϕ and sub-swarm size in APSODEE are set to 20, 0.4 and 2, respectively.

For each training, we run 200 episodes. Here, each episode refers to that all the taxis finish the working in a specific time period (4 hours as introduced above). For each time period, we independently conduct the test for 20 times and record the mean value (AVG) and the standard deviation (SD). Regardless of whether it is workday or weekend, we present the test results.

4.4 Results

In the testing results, the total profit, the total picking up time, the total non-hailing time, and the number of the total orders obtained with different strategies are presented as



(a) Reward curves based on synthetic data set during 8:00-12:00

(b) Reward curves based on synthetic data set during 12:00-16:00

(c) Reward curves based on synthetic data set during 20:00-24:00

(d) Reward curves based on workday data set during 8:00-12:00

(e) Reward curves based on workday data set during 12:00-16:00

(f) Reward curves based on workday data set during 20:00-24:00

(g) Reward curves based on weekend data set during 8:00-12:00

(h) Reward curves based on weekend data set during 12:00-16:00

(i) Reward curves based on weekend data set during 20:00-24:00

Fig. 5 The comparison on training reward

follows. We list the results with respect to the synthetic data set in Table 1. The results show that, in general, the reinforcement methods significantly outperform the roaming strategy in all cases. Furthermore, despite of the slight advantage of DTDE, it performs comparatively to ERL. For the standard deviation of the profit, ERL slightly outperforms DTDE and CTDE.

Furthermore, we investigate the performance of different methods when only workday dataset or weekends dataset are used and the results are given in Tables 2 and 3. ERL and DTDE constantly bring increments of the profit. For CTDE, however, although it outperforms the roaming strategy in some cases, it even performs worse than roaming strategy in some cases. This can be explained by that, DTDE and ERL employ multiple Q-networks, which is able to improve the exploration of the corresponding algorithms. An interesting observation is that the performances of CTDE on workdays and weekends are quit

different: it performs better on workdays. This can be explained by that there are more passengers in workdays due to work, leading to more orders and less difficulty for exploration.

To provide a deeper insight into the characteristics of different algorithms, we present Fig. 5 where the reward curves are recorded (The average results of the 20 independent runs of each algorithm). The x-axis represents the episodes index, while the y-axis records the obtained rewards. It is clear that DTDE and ERL perform similarly with respect to the final training reward. However, CTDE only shows increasing reward in the early stage (60-80 episodes). After that, the CTDE becomes unstable and mostly ends with a low reward after about 150 episodes. The reason is that CTDE only employs one Q-network, therefore, although single DQN can find better actions more quickly, the Q-value expresses the accumulated reward of a certain action. Consequently, it will guide many taxis cruising to the same target grid, and the strong competition will cause a lack of orders for all the taxis in the target grid. As shown in Tables 1, 2 and 3, the number of successful orders for CTDE obviously decreases, which increases the vacant running time.

Furthermore, for the convergence comparison of ERL and DTDE, one can find that ERL significantly outperforms DTDE. This can be explained by that ERL is able to ensure the overall reward of all the taxis, since the evolutionary process aims at selecting the Q-network that performs best at each episode with respect to the profit of all the taxis.

In summary, the comparison of the testing results and the training reward curves provide the following findings: first, the proposed Q-learning based taxi cruising strategy is capable of improving the profit and reducing the vacant run time for taxis; second, the proposed ERL is able to improve the exploration of the Q-learning method and is suitable for multi-agent tasks.

5 Conclusion and future work

This paper constructs a multi-taxi working environment with real order data set and propose a reinforcement learning strategy for multi-taxis cruising. Furthermore, an evolutionary reinforcement learning method is put forward. In the experiment, we demonstrate that the proposed reinforcement learning model for multi-taxis cruising brings a significant improvement in drivers profit. On the other hand, it is notable that the proposed evolutionary reinforcement learning method is demonstrated with an expressive convergence speed, indicating that it is effective in multi-agent tasks. In future, we will further consider multi-objectives, such as optimizing the drivers' path planning, reducing passengers' waiting time, flat-share model for taxis and so forth. We will also aim to utilize other advanced algorithms for multi-agent reinforcement learning in training our strategies.

Acknowledgements This work is supported by the National Natural Science Foundation of China under Grant Number 71771176 and 61503287, Natural Science Foundation of Shanghai, China under Grant Number 19ZR1479000, 20692191200, Fundamental Research Funds for the China Central Universities under Grant Number 22120190202, China Scholarship Council under File Number 201906260030, Shanghai Municipal Science and Technology Major Project (2021SHZDZX0100) and the Fundamental Research Funds for the Central Universities.

Declarations

Conflict of Interest The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with this manuscript submitted.

References

1. Zong F, Wu T, Jia H (2019) Taxi drivers' cruising patternsinsights from taxi gps traces. IEEE Trans Intell Transp Syst 20(2):571–582
2. Ceder A (2011) Public-transport vehicle scheduling with multi vehicle type. In: 18th International Symposium on Transportation and Traffic Theory(ISTTT 18), pp. 485–497. Pergamon-Elsevier Science LTD, Oxford, England
3. Jiang X, Guo HSX, Gong X (2019) Integrated optimization for timetabling and vehicle scheduling of zone urban and rural bus. J Trans Syst Eng Inf Technol 19(3):141–148
4. Bie Y, Tang R, Wang L (2020) Bus scheduling of overlapping routes with multi-vehicle types based on passenger od data. IEEE ACCESS 8:1406–1415
5. Bhattacharya B, Hu Y (2010) Approximation algorithms for the multi-vehicle scheduling problem. In: 21st Annual International Symposium on Algorithms and Computations, pp. 192–205. Springer, Heidelberg, Berlin
6. Kawano H (2010) Applicability of multi-vehicle scheduling problem based on gps tracking records. In: 18th International Conference on Geoinformatics. IEEE, New York, NY, USA
7. Bakas I, Drakoulis R, Floudas N, Lytrivis P, Amditis A (2016) A flexible transportation service for the optimization of a fixed-route public transport network. 6th Transport Research Arena (TRA). Elsevier Science BV, Amsteradm, Netherlands, pp 1689–1698
8. Wei Y, Avci C, Liu J, Belezamo B, Aydin N, Li P, Zhou X (2017) Dynamic programming-based multi-vehicle longitudinal trajectory optimization with simplified car following models. Trans Res Part B-Methodol 106:102–129
9. Chen C, Liu Q, Wang X, Liao C, Zhang D (2021) Semi-traj2-graph: Identifying fine-grained driving style with gps trajectory data via multi-task learning. IEEE Trans Big Data
10. Guo S, Chen C, Wang J, Ding Y, Liu Y, Ke X, Yu Z, Zhang D (2020) A force-directed approach to seeking route recommendation in ride-on-demand service using multi-source urban data. IEEE Trans Mobile Comput
11. Lee D, Wang H, Chen R, Teo S (2004) Taxi dispatch system based on current demands and real-time traffic conditions. In:

83rd Annual Meeting of the Transportation-Research-Board, pp. 193–200. Sage Publications INC, Thousand Oaks, CA
12. Maciejewski M, Bischoff J, Nagel K (2016) An assignment-based approach to efficient real-time city-scale taxi dispatching. IEEE Intell Syst 31(1):68–77
13. Nourinejad M, Ramezani M (2016) Developing a large-scale taxi dispatching system for urban networks. In: IEEE 19th International ZConference on Intelligent Transportation Systems, pp. 441–446. IEEE Press, Hoboken, NJ, USA
14. Wang Y, Liang B, Zheng W, Huang L, Liu H (2016) The development of a smart taxicab scheduling system:a multi-source fusion perspective. In: IEEE 16th International Conference on Data Mining, pp. 1275–1280. IEEE Press, Hoboken, NJ, USA
15. Liu Z, Miwa T, Zeng W, Bell MG, Morikawa T (2018) Shared autonomous taxi system and utilization of collected travel-time information. J Adv Trans 2018
16. Liu L, Andris C, Ratti C (2010) Uncovering cabdrivers behavior patterns from their digital traces. Comput, Environ Urban Syst 34(6):541–548
17. Li B, Zhang D, Sun L, Chen C, Li S, Qi G, Yang Q (2011) Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset. In: 2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 63–68. IEEE
18. Zhang K, Chen Y, Nie YM (2019) Hunting image: Taxi search strategy recognition using sparse subspaceclustering. Trans Res Part C: Emerg Technol 109:250–266
19. Bai R, Li J, Atkin J, Kendall G (2014) A novel approach to independent taxi scheduling problem based on stable matching. J Oper Res Soc 65(10):1501–1510
20. Chen C, Zhang D, Ma X, Guo B, Wang L, Wang Y, Sha E (2016) Crowddeliver: Planning city-wide package delivery paths leveraging the crowd of taxis. IEEE Trans Intell Transp Syst 18(6):1478–1496
21. Seow K, Dang N, Lee D (2010) A collaborative multiagent taxi-dispatch system. IEEE Trans Autom Sci Eng 7(3):607–616
22. Cordeau J, Laporte G (2007) The dial-a-ride problem: models and algorithms. Ann Oper Res 153(1):29–46
23. Attanasio A, Cordeau J, Ghiani G, Laporte G (2004) Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parallel Comput 30(3):377–387
24. Nie J (2015) Research on multi-vehicle scheduling problem based on dynamic demand. International Conference on Intelligent Transportation. Big Data and Smart City (ICITBS). IEEE, New York, NY, USA, pp 657–660
25. Zhu C, Tang L, Zhang W (2014) Multi-vehicle coordination and flexible scheduling based on simulated annealing algorithm. In: 26th Chinese Control and Decision Conference (CCDC), pp. 2686–2691. IEEE, New York, NY, USA
26. Powell JW, Huang Y, Bastani F, Ji M (2011) Towards reducing taxicab cruising time using spatio-temporalprofitability maps. In: Proceedings of the 12th International Conference on Advances in Spatial and Temporal Databases, pp. 242–260. Springer, Heidelberg, Berlin
27. Luo Z, Xie R, Huang W, Shan Y (2017) Intelligent taxi dispatching based on artificial fish swarm algorithm. Syst Eng-Theory Practice 37(11):2938–2947
28. Youqin S, Chao C, Qin Z (2018) Taxi resource allocation optimization under improved particle swarm optimization algorithm based on multi-chaotic strategy. J Heilongjiang Univ Technol 18(5):72–76
29. Xu J, Rahmatizadeh R, Boloni L, Turgut D (2018) Real-time prediction of taxi demand using recurrent neural networks. IEEE Trans Intell Transp Syst 19(8):2572–2581
30. Verma T, Varakantham P, Kraus S, Lau HC (2017) Augmenting decisions of taxi drivers through reinforcementlearning for improving revenues. In: Proceedings of the 27th International Conference on Automated Planningand Scheduling, pp. 409–417. The AAAI Press, Palo Alto, California, USA
31. Rong H, Zhou X, Yang C, Shafiq Z, Liu A (2016) 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
32. Han M, Senellart P, Bressan S, Wu H (2016) 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, NY, USA
33. Shi D, Ding J, Errapotu SM, Yue H, Xu W, Zhou X, Pan M (2019) Deep q-network-based route scheduling for tnc vehicles with passengers location differential privacy. IEEE Int Things J 6(5):7681–7692
34. Yang Y, Wang X, Xu Y, Huang Q (2020) Multiagent reinforcement learning-based taxi predispatching model to balance taxi supply and demand. J Adv Trans 2020
35. Liu C, Chen C-X, Chen C (2021) Meta: A city-wide taxi repositioning framework based on multi-agent reinforcement learning. IEEE Trans Intell Trans Syst
36. Zeng W, Wu M, Sun W, Xie S (2020) Comprehensive review of autonomous taxi dispatching systems. Comput Sci 47(5):181–189
37. Wang Y, Liu H, Zheng W, Xia Y, Li Y, Chen P, Xie H (2019) Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning. IEEE ACCESS 7:29974–39982
38. Tseng HH, Luo Y, Cui S (2017) Deep reinforcement learning for automated radiation adaptation in lung cancer. Med Phys 44(12):6690–6705
39. Carta S, Ferreira A, Podda AS (2021) Multi-dqn: an ensemble of deep q-learning agents for stock market forecasting. Exp Syst Appl 164
40. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction. MIT press, Cambridge, MA
41. Leemon B (1995) Residual algorithms: Reinforcement learning with function approximation. In: Proceedings of the International Conference on Machine Learning, pp. 30–37. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA
42. Watkins H, Dayan P (1992) Q-learning. Mach Learn 8:279–292
43. Justin B, Andrew M (1995) Generalization in reinforcement learning: Safely approximating the value function. In: Advances in Neural Information Processing Systems, pp. 369–376. MIT Press, Cambridge, MA
44. Mnih V, Kavukcuoglu K, Silver D (2015) Human-level control through deep reinforcement learning. Nature 518(7450):529–533
45. Hasselt H (2010) Double q-learning. In: In Advances in Neural Information Processing Systems 23 (NIPS-10), pp. 1613–2621
46. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P (2015) Trust region policy optimization, 1889–1897. PMLR
47. Sun Y, Xue B, Zhang M, Yen GG (2019) Evolving deep convolutional neural networks for image classification. IEEE Trans Evol Comput 24(2):394–407
48. Salimans T, Ho J, Chen X, Sidor S, Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning. arXiv:1703.03864
49. Khadka S, Tumer K (2018) Evolutionary reinforcement learning. arXiv:1805.07917
50. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
51. Pourchot A, Sigaud O (2019) Cem-rl: Combining evolutionary and gradient-based methods for policy search. arXiv:1810.01222 [cs.LG]

52. Parker-Holder J, Pacchiano A, Choromanski K, Roberts S (2020) Effective diversity in population based reinforcement learning. arXiv:2002.00632 [cs.LG]
53. Tan M (1993) Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337
54. Gronauer S, Diepold K (2021) Multi-agent deep reinforcement learning: a survey. Artif Intell Rev 2020
55. Li D, Guo W, Lerch A, Li Y, Wang L, Wu Q (2021) An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization. Swarm Evol Comput 60:100789