Sustainable Cities and Society 89 (2023) 104311

Contents lists available at ScienceDirect

Sustainable Cities and Society

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

Deep reinforcement learning-based long-range autonomous valet parking for smart cities

Muhammad Khalid [a], Liang Wang [b], Kezhi Wang [c], Nauman Aslam [d], Cunhua Pan [e], Yue Cao [f,*]

[a] School of Computer Science, University of Hull, Hull, HU6 7RX, UK
[b] School of Aerospace, Transport and Manufacturing, Cranfield University, Milton Keynes, MK43 0AL, UK
[c] Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UB8 3PH, UK
[d] Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, NE1 8ST, UK
[e] National Mobile Communications Research Laboratory, Southeast University, China
[f] School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

ARTICLE INFO

Keywords:
Long-range autonomous valet parking (LAVP)
Autonomous vehicle
Deep reinforcement learning
Ant colony optimization (ACO)
Sustainable cities and communities

ABSTRACT

In this paper, to reduce the congestion rate at the city center and increase the traveling quality of experience (QoE) of each user, the framework of long-range autonomous valet parking is presented. Here, an Autonomous Vehicle (AV) is deployed to pick up, and drop off users at their required spots, and then drive to the car park around well-organized places of city autonomously. In this framework, we aim to minimize the overall distance of AV, while guarantee all users are served with great QoE, i.e., picking up, and dropping off users at their required spots through optimizing the path planning of the AV and number of serving time slots. To this end, we first present a learning-based algorithm, which is named as Double-Layer Ant Colony Optimization (DLACO) algorithm to solve the above problem in an iterative way. Then, to make the fast decision, while considers the dynamic environment (i.e., the AV may pick up and drop off users from different locations), we further present a deep reinforcement learning-based algorithm, i.e., Deep Q-learning Network (DQN) to solve this problem. Experimental results show that the DL-ACO and DQN-based algorithms both achieve the considerable performance.

1. Introduction

The mobility of urban area is highly dependent on the transportation system. Effective transportation plays an important role in the sustainability and development of future smart cities (Zhu, Shen, & Ren, 2022). Normally, city centers are the busiest places and difficult for the traffic congestion control (Li, Taeihagh, & Tan, 2022; Ni, Lin, & Shen, 2019). A large number of vehicles enter and leave the city center every hour (Gyawali & Qian, 2019; Orejon-Sanchez, Crespo-Garcia, Andres-Diaz, & Gago-Calderon, 2022). It may result in various mobility issues, e.g., high congestion, pollution and fuel consumption rate as well as long journey time for the people. One of the key issues leading to the above problems may be that the vehicles keep searching for Car Park (CP) in city center (Cogill et al., 2014; Heidari, Navimipour, & Unal, 2022). According to the survey, 30% of overall traffic in urban area is caused by drivers searching for a suitable CP (Shoup, 2017). On average, it takes a driver about 6–20 min in UK to find a CP (Shoup, 2006). This is due to the fact that the drivers looking for CP may not have the prior and background information about CP or information about best route (Mackey, Spachos, & Plataniotis, 2020; Vondra, Becvar, & Mach, 2016). To reduce the searching and roaming time, the Smart Parking (SP) has been proposed, which gives drivers an opportunity to receive information of CP on their smart phones. The message provides user with real-time available parking slots in CP (Yue, Abdel-Aty, Wu, & Farid, 2019). Thanks to the Information and Communication Technologies (ICT) as well as machine learning and computer vision based solutions, finding available CP slot becomes more and more convenient (Kirschner & Lanzendorf, 2020; Zhu et al., 2022). However, the SP may suffer from challenging scenarios where each vehicle rushes to the same CP and thus may result in a higher congestion rate in the city center (Kotb, Shen, & Huang, 2017; Zhong, Ni, Cui, Zhang, & Liu, 2021).

In the future, we envision that almost all the Autonomous Vehicle (AV) can operate on battery power and move autonomously, contributing to eco-friendly environment. The AV can reduce the carbon emission and minimize journey cost, when compared with fuel-powered cars (Shaaban, Mohamed, Ismail, Qaraqe, & Serpedin, 2019). In addition, AV is expected to offer Autonomous Valet Parking (AVP),

* Corresponding author.
E-mail address: 871441562@qq.com (Y. Cao).

https://doi.org/10.1016/j.scs.2022.104311
Received 4 June 2022; Received in revised form 26 October 2022; Accepted 18 November 2022
Available online 25 November 2022
2210-6707/© 2022 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

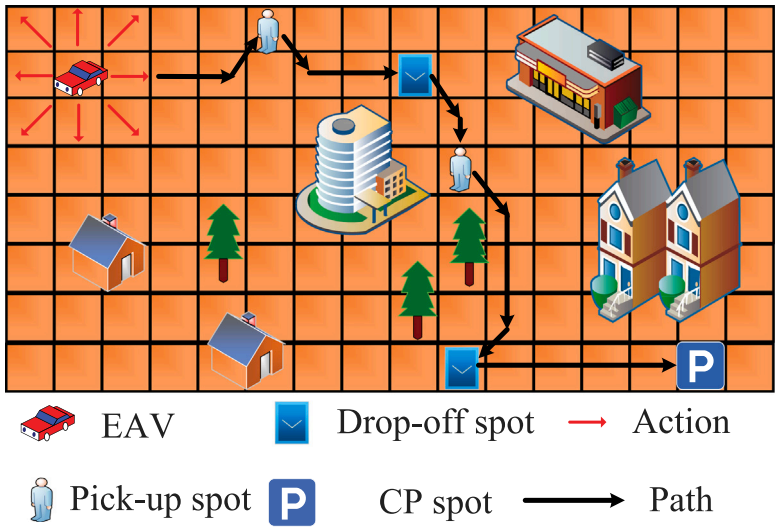


Fig. 1. The proposed long-range autonomous valet parking framework.

which can help the AV find the suitable car park slot in CP. Normally, AVP provides two kinds of solutions: (1) Short-range Autonomous Valet Parking (SAVP); and (2) Long-range Autonomous Valet Parking (LAVP).

For the first case, the user can leave vehicle at the CP entrance. Then SAVP can guide the AV to find a suitable parking spot itself inside CP. Specially, the AVP applies 3-dimensional (3D) localization and computer vision techniques to move between different stories of a CP. It also searches for a vacant parking slot as well as avoid obstacles on their way (Paidi, Fleyeh, Håkansson, & Nyberg, 2018). In addition, users can pick up their AVs at the entrance of CP for the return journey, which may be synchronized with the movement of user to avoid delays as well as congestion inside CP. However, the key issue of SAVP is that the users still have to go a long way to the entrance of CP themselves, probably from their work place or city center via other means, however, as mentioned earlier, this process may take around 6–20 min on average. On the other hand, the LAVP proposes to deploy CP at the border of city center to avoid congestion inside the city center (Khalid et al., 2019). In this case, instead of going to the entrance of CP, the users can leave the AV anywhere. Such as the place close to their company or inside the city center where they can do shopping. These places can be called as the drop-off spots. Once people gets out of the AV, the AV can move to the available CP autonomously controlled by LAVP. For the return journey, users may use their mobile phones to book the AV, which will pick-up them at any spot as set by them, via applying the similar techniques. Normally, the drop-off spots can be set the same as the pick-up spots. To facilitate this process, the advanced vision and optimization techniques can be applied to minimize overall cost (e.g., resources, battery, time) of AVs during the above process (Hodorog, Petri, & Rezgui, 2022).

In above mentioned scenarios, the key challenges are yet to be addressed, i.e., path planning of the AV in dynamic environment, while also consider (1) meeting the quality of experience (QoE) of each user, (2) serving the user with the optimal order, i.e., picking up user first and dropping off it later, (3) minimizing the overall distance of the AV. The above optimization normally includes the mixed integer variables, which is very difficult to tackle. The traditional convex-based solutions, due to high complexity, may not be applied in the above situations. Additionally, the random search does not normally provide optimal solution, as it is difficult to converge due to the non-learning and environment adaptability behavior. Furthermore, other hybrid optimization based solutions, e.g., greedy optimization normally needs several iterations and also suffer from high complexity. If the environment varies, the above mentioned optimizations may have to be re-run to adapt to the new environment, which may not suitable for the fast decision making in the dynamic environment (Singh et al., 2020).

Against the above background, in this paper, we aim to design the long-range autonomous valet parking framework, as shown in Fig. 1. In this framework, we optimize the path planning of AV, aiming at minimizing the overall distance of the AV, while guaranteeing all users are served, by picking up, and dropping off them to their required spots. To this end, we propose two learning based solutions, i.e., Double-Layer Ant Colony Optimization (DL-ACO) and DQN-based algorithms to solve the above problem. The DL-ACO can be applied to the new scenario or unfamiliar environment, especially for the case when DQN is difficult to be trained and converge. The DL-ACO can also be used to verify the results of DQN based algorithm in uncertain scenarios. On the other hand, the DQN can be applied to the familiar environment to make the efficient and fast decisions. This is because we can pre-train DQN and once the training is done, DQN can make decision very fast. The proposed techniques will improve the overall ride sharing and parking experience and it will also contribute towards green environment and sustainable development.

The remainder of this paper is organized as follows. Section 2 presents the related work and Section 3 describes the system model. Section 4 introduces the proposed DL-ACO algorithm, while Section 5 presents the DQN-based algorithm. The simulation results are reported in Section 6, followed by conclusions at last.

2. Literature review

Finding a suitable car parking slot is one of the key challenges in transportation systems. The optimal path planning plays an important for vehicles to search for a suitable car park slot while also minimizing the resources required, e.g., battery consumption (Fan, Zhang, Yu, Hong, & Dong, 2022; Javed & Zeadally, 2018). Popular path planning and route optimization algorithms include Dijkstra, Ant Colony Optimization (ACO) and A-Star (A*) (Dorigo, Birattari, & Stutzle, 2006), which can help to find the shortest path from source to destination. The Dijkstra algorithm divides road into edges and each edge is assigned with the weight. The weight of edges varies from scenario to scenario, which can either be energy, time or distance. In ACO, the artificial ants use swarm intelligence to find the shortest route. The biological ants when searches for food can leave some hormones called pheromone on their way to destination. This pheromone is sensed by other ants and then they can follow the same path. If more ants travel with the same path, there will be a higher amount of pheromone. However, the above-mentioned algorithms do not normally learn from the past

experience. This means if the source or destination changes, they may have to re-solve the problem to get the new optimal solutions.

Machine learning based solution has been proposed recently to assist real-time decision making in various situations. It mainly involves three main subcategories: supervised learning, unsupervised learning and reinforcement learning (Fadlullah et al., 2017). Supervised learning requires labeled data for decision making, whereas unsupervised learning is applied to learn data pattern and relationships from unlabeled or unknown data. The reinforcement learning, on the other hand, can be applied to the environments where no prior information is available. The learning process is achieved through direct interaction with environment. The reinforcement learning normally has five main elements: an agent, environment, state, action and reward (Luong et al., 2019). In reinforcement learning, an agent can interact with the environment by taking actions and obtain the accumulated reward. The reinforcement learning may contain multiple episodes to fully train an agent in the environment. Then the decision making can be done, with the help of the agent. The reinforcement learning technique has shown huge potentials in autonomous vehicles, such as Unmanned Aerial Vehicles (UAVs). In Wang et al. (2019), Wang et al. proposed a RLAA algorithm based on Q-learning to optimize the user association and resource allocation in UAV-enabled MEC. In Wang et al. (2020), a UAV-aided MEC framework is investigated, where a group of UAVs cooperate to serve ground UEs, and the authors developed a MAT algorithm based on multi-agent reinforcement learning to optimize the trajectories of UAVs.

The transportation system of a city is one the most dynamic system where a robust decision handling system is a crucial element. The robust decision system for transportation in smart cities will always contribute towards sustainability and green environment. The deep reinforcement learning after certain amount of training is capable of handling real time scenarios. In this paper, to assist LAVP in the dynamic environment, two learning-based algorithms are proposed. Specifically, the DL-ACO algorithm can be applied in unfamiliar scenarios, whereas DQN-based algorithm can be deployed in dynamic or familiar environment, after adequate training.

3. System model

In this section, we consider the LAVP scenario, as shown in Fig. 1, where the city map is divided into a grid map. We assume that there is an AV serving N users within the city. To simplify this, we formulate the city as a $Z^X \times Z^Y$ grid map, which contains several obstacles, and we define the set of users as $\mathcal{N} \triangleq \{n = 1, 2, \dots, N\}$. The AV starts to serve users from the initial taking-off spot, whose coordinate is denoted as $q^I = [X^I, Y^I]$. Additionally, the AV serves the user n by visiting the pick-up spot, whose coordinate is $q_n^P = [X_n^P, Y_n^P]$, and visiting the drop-off spot, whose coordinate is $q_n^D = [X_n^D, Y_n^D]$. Finally, the AV reaches the CP spot, whose coordinate is denoted as $q^C = [X^C, Y^C]$, after all users are served.

Note that, we assume this process lasts for T time slots or steps, which vary with the path planning of AV. For simplicity, we denote the set of time slots as $\mathcal{T} \triangleq \{t = 1, 2, \dots, T\}$, and in the time slot t, the AV will select an action $a(t)$ from the action set, which is denoted as $\mathcal{A} \triangleq$ {UP, DOWN, LEFT, RIGHT, TOP-LEFT, TOP-RIGHT, BOTTOM-LEFT, BOTTOM-RIGHT}. Thus, it has

$$a(t) \in \mathcal{A}, \forall t \in \{1, 2, \dots, T\}. \tag{1}$$

Then, given the coordinate of the AV, denoted by $q(t) = [X(t), Y(t)]$ in time slot t, the coordinate of the AV $q(t+1) = [X(t+1), Y(t+1)]$ in next time slot is defined as follows:

$$q(t+1) = \begin{cases} [X(t)-1, Y(t)], & \text{if UP} \\ [X(t)+1, Y(t)], & \text{if DOWN} \\ [X(t), Y(t)-1], & \text{if LEFT} \\ [X(t), Y(t)+1], & \text{if RIGHT} \\ [X(t)-1, Y(t)-1], & \text{if TOP-LEFT} \\ [X(t)-1, Y(t)+1], & \text{if TOP-RIGHT} \\ [X(t)+1, Y(t)-1], & \text{if BOTTOM-LEFT} \\ [X(t)+1, Y(t)+1], & \text{if BOTTOM-RIGHT} \end{cases} \tag{2}$$

Thus, the distance traversed between t and $t-1$ is expressed as

$$d(t, t-1) = \sqrt{\|q(t) - q(t-1)\|^2}, \ \forall t \in \mathcal{T}, \tag{3}$$

where $\|\cdot\|$ denotes Euclidean norm. Also, the AV can only move to its adjacent grid within the target grid map in each time slot. It has:

$$0 \le X(t) \le Z^X, \ \forall t \in \mathcal{T}, \tag{4}$$

and

$$0 \le Y(t) \le Z^Y, \ \forall t \in \mathcal{T}. \tag{5}$$

In each time slot, the AV will also need to avoid obstacles, and it has

$$q(t) \notin \mathcal{O}, \forall t \in \mathcal{T}, \tag{6}$$

where $\mathcal{O}$ is the set of the coordinates of obstacles.

In our paper, the traveling of AV is associated with pick-up spot, whose coordinate is $q_n^P = [X_n^P, Y_n^P]$, and a drop-off spot, whose coordinate can be expressed as $q_n^D = [X_n^D, Y_n^D]$. For simplicity, we assume if the AV reaches the pick-up spot where user n is present, the user is picked by the AV, which can be expressed as $q(t) = q_n^P$, i.e., $[X(t), Y(t)] = [X_n^P, Y_n^P]$. Similarly, if the AV reaches the drop-off spot, it drops that user n at that spot, which can be defined as $q(t) = q_n^D$, i.e., $[X(t), Y(t)] = [X_n^D, Y_n^D]$. Without loss of generality, we introduce the set $U(t)$ to present the serving status of all users in time slot t as follows:

$$U(t) = \{u_n(t), \forall n \in \mathcal{N}\}, \forall t \in \{1, 2, \dots, T\}, \tag{7}$$

where $u_n(t)$ is the serving status of user n in time slot t, which has

$$u_n(t) = \begin{cases} 0, & \text{if } q_n^P \notin \{q(t'), t' = 1, 2, \dots, t\}, \\ & \text{and } q_n^D \notin \{q(t'), t' = 1, 2, \dots, t\}, \\ 1, & \text{if } q_n^P \in \{q(t'), t' = 1, 2, \dots, t\}, \\ & \text{and } q_n^D \notin \{q(t'), t' = 1, 2, \dots, t\}, \\ 2, & \text{if } q_n^P \in \{q(t_1), t_1 = 1, 2, \dots, t\}, \\ & \text{and } q_n^D \in \{q(t_2), t_2 = 1, 2, \dots, t\}, t_1 < t_2, \end{cases} \tag{8}$$

in which, from the first time slot to the current time slot t, (1) if the AV does not reach the pick-up and drop-off spot, $u_n(t) = 0$; (2) if the AV reaches the pick-up spot, but does not reach the drop-off spot, $u_n(t) = 1$; and (3) if the AV first reaches the pick-up spot, and then arrives the drop-off spot, $u_n(t) = 2$.

After all the users are served, the AV then reaches the CP spot, and serving process terminates. The serving process terminates. Thus, we can have:

$$\sum_{n=1}^{N} u_n(t) = 2N, t = T, \tag{9}$$

and

$$q(t) = q^C, t = T. \tag{10}$$

Besides, the AV will start serving users from the initial taking off spot. Thus, it has:

$$q(t) = q^I, t = 0. \tag{11}$$

We aim to minimize the overall distance for the AV, while at the same time to make sure all users are served, through optimizing the path planning of the AV. To this end, we formulate our problem as follows:

$$\begin{aligned} &\mathcal{P}: \min_{\boldsymbol{q},\boldsymbol{T}} \sum_{t=1}^{T} d(t,t-1) \\ &\text{subject to constraints}: \\ &(1),(4),(5),(6),(9),(10),(11), \end{aligned} \tag{12a}$$

in which, $\boldsymbol{q} = \{q(t), \forall t \in \mathcal{T}\}$, and $\boldsymbol{T} = \{1, 2, \ldots, T\}$.

The above optimization $\mathcal{P}$ is quite challenging to tackle as one has to decide the optimal path planning, i.e., $\boldsymbol{q}$ and the minimal number of time slots, i.e., $\boldsymbol{T}$ that the AV moves. Precisely, the AV has to avoid obstacles in each time slot, and it must serve users in the adequate order. That is to say each of user should be picked up first and then dropped off next to the adequate spot. After that the AV has to reach the CP spot.

Next, we will propose two solutions to address the above problem. Firstly, we propose a learning based DL-ACO algorithm, which can solve the problem in an iterative way. The DL-ACO algorithm can be applied in some unfamiliar scenarios. Then, to adapt to the dynamic environment, we further present a DQN-based algorithm, which can achieve the solutions in a fast way, once the training process is done.

4. Proposed DL-ACO algorithm

The ACO can be applied in different optimization scenarios such as graph traversing, job scheduling and traveling salesman problems (Kocsány & Szádeczky-Kardoss, 2022; Pacini, Mateos, & Garino, 2014). The basic idea is to apply the biological ants working in form of a group to find a global optimal path. This path is between their nest and destination. The biological ants can leave a special kind of chemical called pheromone when they search for the destination. Then, the following ants will search according to pheromone left on the ground. Besides, the density of pheromone is related to the overall distance between the nest and destination. Thus, if the density of pheromone on a particular path is the highest among others, Then, most ants choose this path, which is the shortest path.

In our paper, the DL-ACO algorithm consists of two parts. Specifically, we first apply it to achieve the best path planning between each pair of spots, including taking-off, pick-up, drop-off, and CP spot. Then, we further apply it to find the optimal order for serving users. The details are given as follows.

Algorithm 1 Path planning between different spots

1: Initialize the distance matrix $\boldsymbol{D^m}$ with size $2N+2$;
2: Establish the spot list E^s with size $2N+2$;
3: **for** spot $i \in E^s$ **do**
4: **for** spot $j \in E^s$ **do**
5: Initialize pheromone matrix $\boldsymbol{p^m}$ with size Z;
6: Initialize vector D^e;
7: **for** iteration $l = 1, 2, \ldots e^{max}$ **do**
8: **for** ant $k = 1, 2, \ldots k^{max}$ **do**
9: **for** $t = 1, 2, \ldots, T^{max}$ **do**
10: Select an action from $\mathcal{A}$ with probability $p^k_{v,w}$;
11: **end for**
12: **if** ant k reaches spot j from spot i **then**
13: Calculate overall distance d^a_k according to (3);
14: $D^e \leftarrow d^a_k$;
15: Update pheromone matrix $\boldsymbol{p^m}$;
16: **end if**
17: **end for**
18: **end for**
19: $\boldsymbol{D^m}_{i,j} = \min(D^e)$;
20: **end for**
21: **end for**

Motivated by the work in Liu, Yang, Liu, Tian, and Gao (2017), the first part of the proposed DL-ACO is described in the Algorithm 1. Specifically, we first define a distance matrix $\boldsymbol{D^m}$ with size $2N+2$, which is used to store the minimal distance between each pair of spots, as shown at Line 1. Besides, we establish a spot list E^s with size $2N+2$ that the AV must reach. Then, from Line 3, we start to find the minimal distance value between spot i and spot j. From Line 5 to 6, we initialize the pheromone matrix $\boldsymbol{p^m}$ with size Z, and we define a vector D^e to store the successful distance value achieved by each ant. Then, from Line 10, the ant k starts to explore and selects an action from $\mathcal{A}$. Note that the ant k moves from the current grid v to the next available grid w with the probability $p^k_{v,w}$, expressed as follows:

$$p^k_{v,w} = \frac{\tau^{\alpha}_{v,w}\eta^{\beta}_{v,w}}{\sum_{z\in \text{allowed}_v} \tau^{\alpha}_{v,z}\eta^{\beta}_{v,z}}, \tag{13}$$

where $\tau_{v,w}$, $\eta_{v,w}$ are the amount of pheromone left and the attractiveness from grid v to grid w, $\eta_{v,w}$. α, β are influence parameters which determine the importance of pheromone versus heuristic information.

Then, from Line 12, if the ant k reaches the spot j, the overall distance d^a_k between the pair of spots can be obtained by (3), and will be stored in D^e. Additionally, according to the grids that the ant k visits, the pheromone matrix $\boldsymbol{p^m}$ will be updated given the following equation

$$\tau_{v,w} \leftarrow (1-\rho)\tau_{v,w} + \sum_k \triangle\, \tau^k_{v,w}, \tag{14}$$

where ρ denotes the pheromone decay and $\triangle\, \tau^k_{v,w}$ can be obtained by

$$\triangle\, \tau^k_{v,w} = \begin{cases} \frac{\mu}{d^a_k}, \text{if ant } k \text{ visits path between grid } v,\ w, \\ 0, \text{otherwise} \end{cases} \tag{15}$$

where μ is the constant value.

Furthermore, from Line 19, after adequate iteration, we select the minimal distance value from D^e and store it in $\boldsymbol{D^m}$. Then the best path planning between spot i and spot j is achieved.

Algorithm 2 DL-ACO algorithm for LAVP

1: Obtain the distance matrix $\boldsymbol{D^m}$ from Algorithm 1;
2: Initialize the pheromone matrix $\boldsymbol{p^s}$ with size $2N+2$;
3: Initialize the minimal distance value d^{min};
4: **for** iteration l = 1,2,..., l^{max} **do**
5: **for** ant k = 1,2,...k^{max} **do**
6: Select spots from spot list E^s;
7: Obtain overall distance d_k according to $\boldsymbol{D^m}$;
8: **if** (9), (10), (11) are met and $d_k < d^{min}$ **then**
9: Update pheromone matrix $\boldsymbol{p^s}$;
10: $d^{min} \leftarrow d_k$;
11: **end if**
12: **end for**
13: **end for**
14: Obtain optimal path planning according to d^{min}.

Next, having obtained the distance matrix $\boldsymbol{D^m}$ that can represent the minimal distance between each pair of spots, we further achieve the optimal order for serving users. Precisely, we show the overall algorithm in Algorithm 2. From Line 2 to 3, we initialize the pheromone matrix $\boldsymbol{p^s}$ with size $2N+2$. Besides, we temporarily define a value d^{min} to represent the minimal overall distance value achieved by the ants. Then, from Line 4, each of ant starts to explore and selects the spot that it will visit. Specifically, the ant k moves from the current spot i to the next available spot j with the probability $p^k_{i,j}$.

$$p^k_{i,j} = \frac{\tau^{\alpha}_{i,j}\eta^{\beta}_{i,j}}{\sum_{z\in \text{allowed}_i} \tau^{\alpha}_{i,z}\eta^{\beta}_{i,z}}. \tag{16}$$

After all spots are visited, we obtain the overall distance d_k. If the order of spots that the ant k visits can meet requirement of (9), (10), (11), and in the same time $d_k < d^{min}$ is met, we update the pheromone matrix p^s as follows:

$$\tau_{i,j} \leftarrow (1-\rho)\tau_{i,j} + \sum_k \triangle \tau_{i,j}^k, \tag{17}$$

where $\triangle \tau_{i,j}^k$ is

$$\triangle \tau_{i,j}^k = \begin{cases} \frac{\mu}{d_k^a}, \text{if ant } k \text{ visits path between spot } i,\ j, \\ 0, \text{otherwise.} \end{cases} \tag{18}$$

Then, we set d_k as the temporarily minimal distance value d^{min}. After adequate iteration, we can obtain the optimal path planning according to d^{min}.

5. DQN-based algorithm

The above DL-ACO algorithm may not be suitable for dynamic environment (where pick-up, drop-off and route will change in every episode) as it needs to iterate to find the optimal solutions, and re-run if the locations of pick-up, drop-off spots changes at each cycle. Motivated by this, we introduce a DQN-based algorithm that can achieve the best solutions in a short time. Next, we first introduce some background knowledge of deep reinforcement learning, including deep neural network (DNN), Q-value, and other fundamental elements.

5.1. Background knowledge

The reinforcement learning being an emerging technology playing a beneficial role in scenarios where environment is dynamic and keep changing frequently (Silver et al., 2018). A general reinforcement learning model consists of an agent, action list, reward, states and environment (Silver et al., 2016). Specifically, in the structure of reinforcement learning, an agent is considered to interact with the environment. The process of the interaction can be expressed as the finite Markov decision process (MDP). Specifically, given a series of states $s(t)$, the goal of the agent is to select actions $a(t)$ that can maximize the accumulated rewards $\sum_{t'=t}^{T} \gamma^{t'-t} r(t')$, where γ is the discount factor that balances the immediate and future reward, $r(t)$ is the reward. Additionally, in order to map the relationship between state $s(t)$ and action $a(t)$, an action-value function, it is also known as Q-value $Q(s(t), a(t))$ is defined, which can be expressed as an Bellman equation. Besides, in order to obtain the accumulated rewards, another important element of reinforcement learning named Q-table (Watkins & Dayan, 1992) is applied to store the Q-value of each pair of action and state.

However, the classical reinforcement learning may suffer from high-dimensional space of states and actions as the size of Q-table is finite. Motivated by the development of deep neural networks (DNNs), Mnih et al. (2013, 2015) combined reinforcement learning and DNNs, i.e., DQN, which can replace the Q-table. Additionally, in order to further stabilize the training process, two mechanisms are proposed. First, the memory named experience replay is used to store the experiences of the past, which eases the correlation between each of states. When the action is generated by the DQN, the agent sends the action to the environment, and the state will transfer to the next state. Then, the experience, which consists of $[s(t), a(t), r(t), s(t+1)]$ will be stored in the experience replay memory. When the learning process starts, several experiences will be sampled for training the DQN. The second mechanism is called target network, which has the same structure as the DQN, but it only updates with certain intervals.

5.2. DQN-based algorithm for LAVP

In order to apply DQN-based algorithm in the LAVP framework, we define the action, state, and reward function as follows:

- **Action:** In our proposed scheme, we define $a(t)$ in each time slot as the action of the AV, and it has

$$a(t) \in \mathcal{A}. \tag{19}$$

- **State:** In our paper, the state $s(t)$ consists of the following components:
  - the current coordinate of the AV: $[X(t), Y(t)]$.
  - the coordinates of all pick-up spots: $[X_n^P, Y_n^P], \forall n \in \mathcal{N}$.
  - the coordinates of all drop-off spots: $[X_n^D, Y_n^D], \forall n \in \mathcal{N}$.
  - the coordinate of CP spot: $[X^C, Y^C]$.
  - the serving status of each user: $u_n(t), \forall n \in \mathcal{N}$.

- **Reward:**
  In order to achieve a better performance in terms of convergence, we define the reward function $r(t)$ as follows:

$$r(t) = \begin{cases} -p, \text{ if (4)–(6) are not met} \\ 2p, \text{ if } q(t) = q_n^P \text{ and } u_n(t) = 1 \text{ in the first time} \\ 4p, \text{ if } q(t) = q_n^D \text{ and } u_n(t) = 2 \text{ in the first time} \\ 10p, \text{ if } q(t) = q^C \text{ and } \sum_{n=1}^{N} u_n(t) = 2N \\ -d(t-1, t), \quad \text{otherwise,} \end{cases} \tag{20}$$

  From above, it is observed that (1) if the AV crosses the border or hits the obstacle, the agent will obtain a reward, which is $-p$; (2) if the AV reaches the pick-up spot, i.e., $u_n(t) = 1$, and $q(t) = q_n^P$ in the first time, the reward is $2p$; (3) if the AV reaches the drop-off spot of user n and reached the pick-up spot of user n before, that is to say $u_n(t) = 2$, and $q(t) = q_n^D$ the agent will also receive a reward of $4p$; (4) if the AV reaches the CP spot and all users are served, which means (9), (10), (11) are met, the reward is $10p$; and (5) otherwise the reward is defined as the minus of distance $d(t-1, t)$.

We provide the overall structure of DQN-based algorithm for LAVP in Fig. 2. More precisely, the agent is deployed to control the AV through interacting with the environment, the agent sends the state $s(t)$ to the DQN named evaluation network, which generates the Q-values of all actions. Note that in order to avoid the local optimum, an ϵ-greedy policy is applied. Then, the agent sends the action $a(t)$ to the environment and obtains the reward $r(t)$. After that, the environment transfers to the state $s(t+1)$ of the next time slot. The experience, which consists of $[s(t), a(t), r(t), s(t+1)]$, is stored in the experience replay memory $\boldsymbol{M}$. When the learning process starts, K experiences will be randomly sampled for obtaining the target value, which can be expressed as

$$Q^*(s(t), a(t)) = \mathbb{E}\left[r(t) + \gamma \max_{a(t+1)} Q^*(s(t+1), a(t+1); \theta^-)\right], \tag{21}$$

where $\gamma \in [0, 1]$ is the discount factor Q^* is generated by the target network, θ^- denotes the network parameter.

Then, the loss function can be calculated as follows:

$$L(\theta) = \mathbb{E}\left[\left(Q^*(s(t), a(t)) - Q(s(t), a(t); \theta)\right)^2\right]. \tag{22}$$

The evaluation network is updated according to the following equation:

$$\nabla_\theta L(\theta) = \mathbb{E}\left[\left(Q^*(s(t), a(t)) - Q(s(t), a(t); \theta)\right) \nabla Q(s(t), a(t); \theta)\right]. \tag{23}$$

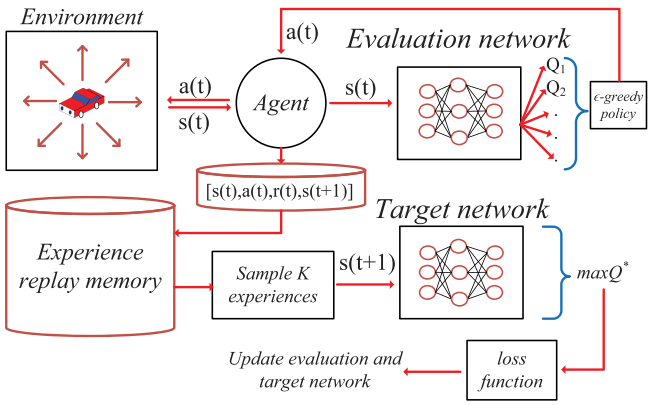


Fig. 2. Overall structure of DQN for LAVP.

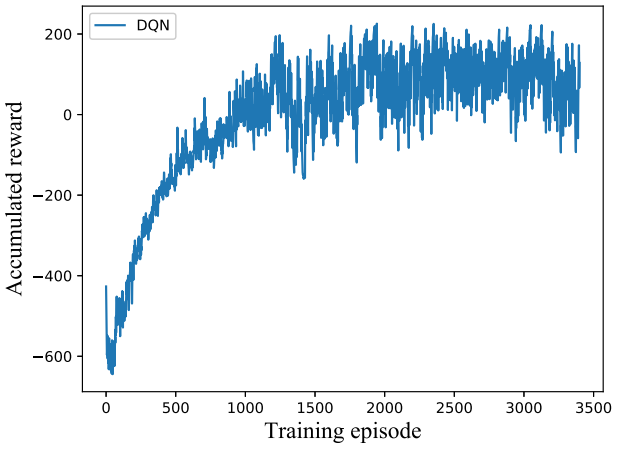


Fig. 3. The convergence performance of DQN.

Algorithm 3 DQN-based algorithm for LAVP

1: Initialize evaluation and target network with parameter θ, θ^{-};
2: Initialize experience replay memory $\boldsymbol{M}$;
3: **for** $E = 1, 2, ..., E^{max}$ **do**
4: Initialize state $s(t)$;
5: **for** $t = 1, 2, ...T^{max}$ **do**
6: Obtain state $s(t)$ from the environment;
7: Select action $a(t)$ that has the largest Q-value with probability ϵ;
8: Randomly select action from $\mathcal{A}$ with probability $1 - \epsilon$;
9: Obtain reward according to (20);
10: Obtain the state of next time slot $s(t+1)$;
11: Store experience $[s(t), a(t), r(t), s(t+1)]$ in memory;
12: **if** learning process starts **then**
13: Randomly sample K experiences from memory;
14: Calculate loss function according to (22);
15: Train the evaluation network according to (23) with learning rate ϕ;
16: Update parameters of the target network: $\theta^{-} = \tau\theta^{-} + (1 - \tau)$;
17: **end if**
18: **end for**
19: **end for**

Table 1
Simulation parameters.

Parameter	Description	Parameter	Description
Z^X	20	Z^Y	20
q^i	[0, 0]	q^c	[19, 19]
N	3	α	1.1
β	12	ρ	0.5
μ	10	e^{max}	10
k^{max}	20	l^{max}	50
ϕ	0.0003	γ	0.99
p	10	ϵ	0.9
τ	0.001	E^{max}	10^6
T^{max}	100	K	256

Furthermore, we show the overall algorithm design in Algorithm 3, from which, we first initialize the evaluation network, and target network with parameters θ, θ^{-} respectively at Line 1. The experience replay memory $\boldsymbol{M}$ is also initialized at Line 2. Then, in each training episode, we initialize the state in the first time slot. The agent interacts with the environment given the state $s(t)$ and receive the action from the evaluation network. Note that for avoiding the local optimum, an ϵ-greedy policy is applied, which means the agent can obtain the action that has the largest Q-value with probability ϵ, or randomly obtains the action from $\mathcal{A}$ with probability $1 - \epsilon$. From Line 9, it obtains the reward according to (20), and transfers to the state of next time slot $s(t+1)$. Then, the experience $[s(t), a(t), r(t), s(t+1)]$ is stored into the experience replay memory. From Line 12, if the learning process starts, K experiences will be randomly sampled for training the networks. Specifically, we obtain the loss value from (22), and train the evaluation network from (23). Additionally, the target network will be updated with the rate τ.

6. Performance analysis

The simulation has been carried out in Python 3.7, tensorflow 1.15.0, INTEL 3450T, and NVIDIA GTX 1050Ti. We divide the simulation area as a 20×20 grid map, which consists of various roads, obstacles and buildings. The AV always starts to serve users from the initial taking off spot, whose coordinate is $q^i = [0, 0]$. After serving all users, the AV will move to the CP spot, whose coordinate is $q^c = [19, 19]$. Once the DQN training is completed, the AV can move to CP anytime. For simulation and analysis purposes, we instruct the AV to park at CP once all passengers are served. In our DQN-based algorithm, we deploy three fully-connected layers with $[400, 300, 300]$ neurons. The AdamOptimizer (Kingma & Ba, 2014) is used. The network is trained with the learning rate $\phi = 0.003$, the target network is updated with the rate $\tau = 0.001$, and the size of experience replay memory is 10^6. More parameters can be found in Table 1.

We first analyze the convergence performance of DQN-based algorithm during the training process in Fig. 3, where there are 3 users waiting to be served by the AV. As shown in Fig. 3, we observe that the accumulated reward achieved by DQN remains at -600 at the beginning. The plausible explanation is that the neural network is not convergent and the AV always moves out of the grid map, which means the penalty is always incurred. After that, the accumulated reward increases rapidly, which means the network starts to converge. Then, after 1000 training episodes, the curve remains between 0 and 200 eventually, which means the DQN-based algorithm has achieved the optimal path planning.

Then, we depict the path planning of AV achieved by the proposed DL-ACO algorithm in Fig. 4. Note that the black rectangle represents obstacle, the gray rectangle represents the spot. In addition, IS represents the initial taking off spot, PS denotes pick-up spot, DS is drop-off spot, CP means CP spot, and red line is the path planning of AV. In Fig. 4(a), there are 3 users, the coordinates of their pick-up and drop-off spots are $q_n^p = \{[3, 4], [7, 9], [10, 5]\}$, $q_n^d = \{[14, 7], [17, 16], [15, 12]\}$ respectively. One can observe that the AV controlled by DL-ACO algorithm serves all users with the order $IS \rightarrow PS1 \rightarrow PS2 \rightarrow PS3 \rightarrow DS1 \rightarrow DS3 \rightarrow DS2 \rightarrow CP$. Additionally, we can also observe that in each pair of spots, the DL-ACO always achieved the shortest distance.

Table 2
Executed time of DL-ACO and DQN.

Case	DL-ACO		DQN		
	Execution	Distance	Training	Testing	Distance
1	112.99 (s)	38.971		1.44 (s)	38.971
2	134.55 (s)	39.556		2.40 (s)	39.556
3	203.66 (s)	53.213	1687.95 (s)	1.04 (s)	46.627
4	128.77 (s)	43.799		0.93 (s)	40.385
5	138.35 (s)	38.971		1.52 (s)	38.385

Then, in Fig. 4(b), we deploy another 3 users, whose coordinates of pick-up and drop-off spots are $q_n^p = \{[0,6],[5,6],[10,8]\}$, $q_n^d = \{[15,6],[15,13],[19,17]\}$ respectively. We can observe that the AV controlled by DL-ACO algorithm serves the users with the order $IS \rightarrow PS1 \rightarrow PS2 \rightarrow PS3 \rightarrow DS1 \rightarrow DS2 \rightarrow DS3 \rightarrow CP$. Besides, we can conclude that DL-ACO still achieved the shortest distance in each pair of spots.

Furthermore, we show the path planning of AV achieved by DL-ACO in another scenario, where the coordinates of users' pick-up and drop-off spots are $q_n^p = \{[4,3],[13,7],[18,6]\}$, and $q_n^d = \{[14,11],[12,16],[16,16]\}$. It is observed that the AV serves the users with the order $IS \rightarrow PS1 \rightarrow PS2 \rightarrow DS1 \rightarrow PS3 \rightarrow DS3 \rightarrow DS2 \rightarrow CP$.

Then, after adequate training, we save the network parameters of DQN for testing. We depict the path planning of AV controlled by DQN in Fig. 5. Note that the spots in Fig. 5 are the same as Fig. 4. As shown in Fig. 4(a), the AV serves the users with the order $IS \rightarrow PS1 \rightarrow PS2 \rightarrow PS3 \rightarrow DS1 \rightarrow DS3 \rightarrow DS2 \rightarrow CP$. The overall distance is the same as Fig. 4(a), although their path planning is slightly different.

In Fig. 4(b), the AV serves the users with the order $IS \rightarrow PS1 \rightarrow PS2 \rightarrow PS3 \rightarrow DS1 \rightarrow DS2 \rightarrow DS3 \rightarrow CP$, which is the same as Fig. 4(b).

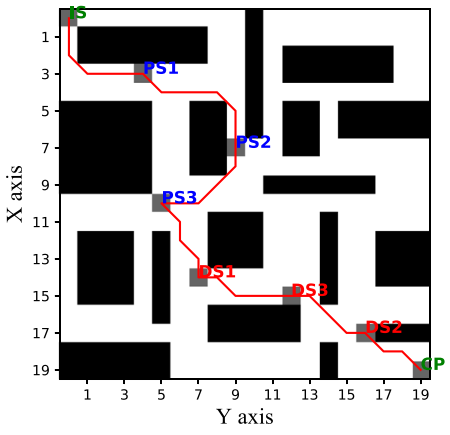
Then, as shown in Fig. 5(c), the AV serves the users with the order $IS \rightarrow PS1 \rightarrow PS2 \rightarrow PS3 \rightarrow DS1 \rightarrow DS2 \rightarrow DS3 \rightarrow CP$. Additionally, compared with the path planning in Fig. 4(c), DQN is much better as its' order for serving users is more practical, which will reduce unnecessary path.

Furthermore, we compare the execution time and distance achieved by DL-ACO, DQN in different cases, i.e., different locations of pick-up, drop-off spots, where there are 3 users. For DQN, the training episodes is 3500. In Table 2, we observe that no matter which case is, DQN always outperforms DL-ACO in terms of over distance. However, it is easy to see that DL-ACO consumes at least 100 s for achieving the considerable performance, while DQN only needs about 1 s in testing, although it takes longer time in training process.

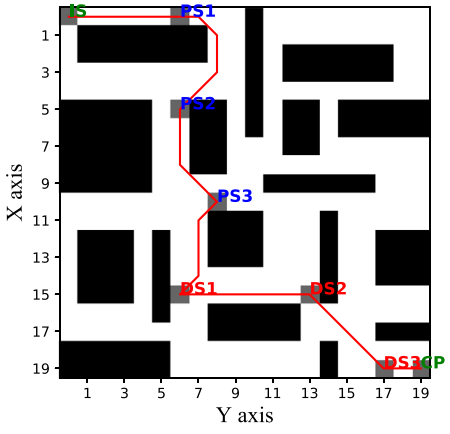
Finally, we analyze the performance of DL-ACO and DQN-based algorithms given different number of users. For fairness, we also apply another random algorithm as comparison. In this setting, the AV randomly selects a available action to take until all users are served and the AV reaches the CP spot. We obtain 500 pairs of path planning of AV achieved by Random and select the optimal result to compare with the proposed DL-ACO and DQN-based algorithms in Fig. 6, from which, we can observe that when the number of users are 1, 2, 4, and 5, our proposed DL-ACO and DQN achieve the similar performance in terms of overall distance. However, when the number of users is 3, DQN outperforms DL-ACO. One plausible explanation is that DL-ACO does not find the best order for serving users. The Random algorithm always performs the worst, no matter how many number of users is.

Conclusions

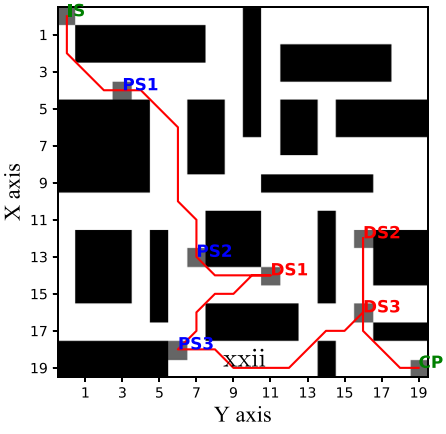
In this paper, we have presented a LAVP framework and shown two learning based solutions, i.e., DL-ACO and DQN-based algorithms for minimizing the overall distance of the AV. The DL-ACO can be applied to the scenarios where pre-training is not required. The limitation here is that decision may take longer time and consume higher



(a)

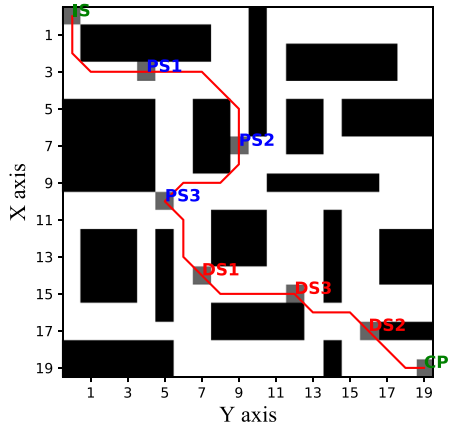


(b)

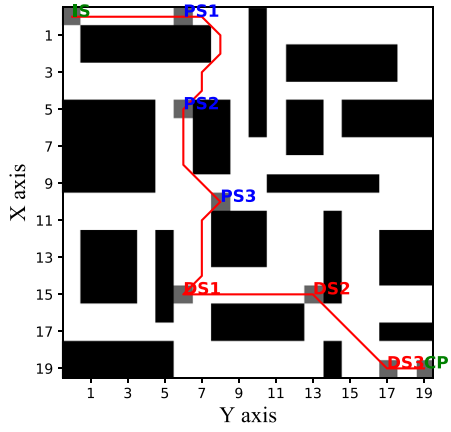


(c)

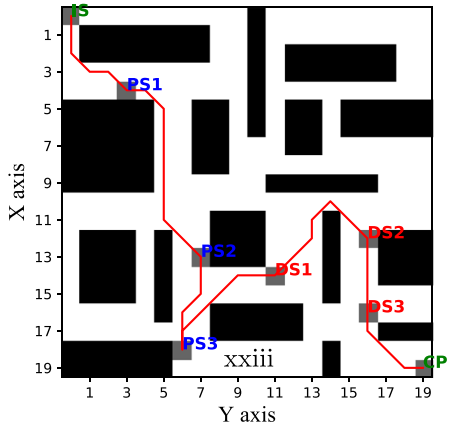
Fig. 4. The path planning of AV achieved by DL-ACO (IS for Initial Spot, PS for Pick-up Spot, DS for Drop-off Spot, CP for CP Spot).



(a)



(b)



(c)

Fig. 5. The path planning of AV achieved by DQN (IS for Initial Spot, PS for Pick-up Spot, DS for Drop-off Spot, CP for Car Parking Spot).

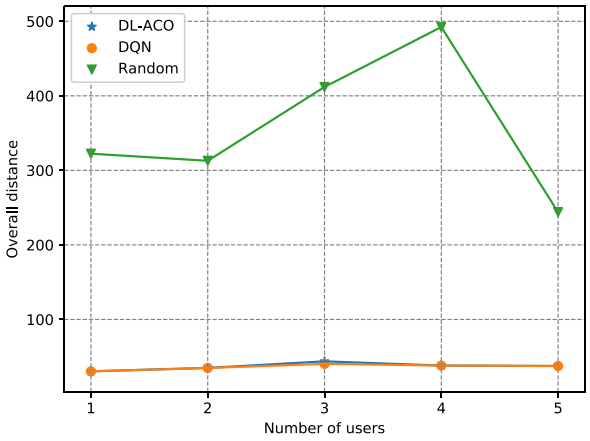


Fig. 6. The overall distance of AV achieved by DL-ACO, DQN and Random given different number of users.

power as compared to the DQN-based algorithm but the time spent on training will not be needed. Similarly, DL-ACO cannot be applied to critical scenarios where a quicker decision is required. The DQN-based algorithm needs training but after the training, it can make faster decision than DL-ACO algorithm. The DQN-based algorithm can be directly applied to critical scenarios. The experimental results have shown that both DL-ACO and DQN-based algorithms can achieve the considerable performance. The proposed techniques may contribute to the decrease of the overall waiting time for parking and improve the experience of people in busy areas, e.g., city center. The DL-ACO and DQN-based algorithms may be readily extended to other urban transportation problems like electric vehicle charging, charging slot deployment and ride sharing schemes.

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.

Data availability

No data was used for the research described in the article.

Acknowledgment

Funding: Wuhan Key Research and Development Program (2022012202015016)

References

Cogill, R., Gallay, O., Griggs, W., Lee, C., Nabi, Z., Ordonez, R., et al. (2014). Parked cars as a service delivery platform. In *2014 international conference on connected vehicles and expo* ICCVE, (pp. 138–143). IEEE.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39.

Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., et al. (2017). State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Communications Surveys & Tutorials*, *19*(4), 2432–2455.

Fan, G.-F., Zhang, L.-Z., Yu, M., Hong, W.-C., & Dong, S.-Q. (2022). Applications of random forest in multivariable response surface for short-term load forecasting. *International Journal of Electrical Power & Energy Systems*, *139*, Article 108073.

Gyawali, S., & Qian, Y. (2019). Misbehavior detection using machine learning in vehicular communication networks. In *ICC 2019-2019 IEEE international conference on communications* ICC, (pp. 1–6). IEEE.

Heidari, A., Navimipour, N. J., & Unal, M. (2022). Applications of ML/DL in the management of smart cities and societies based on new trends in information technologies: A systematic literature review. *Sustainable Cities and Society*, Article 104089.

Hodorog, A., Petri, I., & Rezgui, Y. (2022). Machine learning and Natural Language Processing of social media data for event detection in smart cities. *Sustainable Cities and Society*, *85*, Article 104026.

Javed, M., & Zeadally, S. (2018). RepGuide: Reputation-based route guidance using Internet of Vehicles. *IEEE Communications Standards Magazine*, *2*(4), 81–87.

Khalid, M., Cao, Y., Aslam, N., Raza, M., Moon, A., & Zhou, H. (2019). AVPark: Reservation and cost optimization-based cyber-physical system for long-range autonomous valet parking (L-AVP). *IEEE Access*, *7*, 114141–114153.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kirschner, F., & Lanzendorf, M. (2020). Parking management for promoting sustainable transport in urban neighbourhoods. A review of existing policies and challenges from a German perspective. *Transport Reviews*, *40*(1), 54–75.

Kocsány, L., & Szádeczky-Kardoss, E. G. (2022). Application of mixed graph traversal optimization for the vehicle routing problem. In *2022 European control conference* ECC, (pp. 2149–2154). IEEE.

Kotb, A. O., Shen, Y.-c., & Huang, Y. (2017). Smart parking guidance, monitoring and reservations: a review. *IEEE Intelligent Transportation Systems Magazine*, *9*(2), 6–16.

Li, L., Taeihagh, A., & Tan, S. Y. (2022). What factors drive policy transfer in smart city development? Insights from a Delphi study. *Sustainable Cities and Society*, *84*, Article 104008.

Liu, J., Yang, J., Liu, H., Tian, X., & Gao, M. (2017). An improved ant colony algorithm for robot path planning. *Soft Computing*, *21*(19), 5829–5839.

Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., et al. (2019). Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, *21*(4), 3133–3174.

Mackey, A., Spachos, P., & Plataniotis, K. N. (2020). Smart parking system based on bluetooth low energy beacons with particle filtering. *IEEE Systems Journal*, *14*(3), 3371–3382.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Ni, J., Lin, X., & Shen, X. (2019). Toward privacy-preserving valet parking in autonomous driving era. *IEEE Transactions on Vehicular Technology*, *68*(3), 2893–2905.

Orejon-Sanchez, R. D., Crespo-Garcia, D., Andres-Diaz, J. R., & Gago-Calderon, A. (2022). Smart cities'development in Spain: A comparison of technical and social indicators with reference to European cities. *Sustainable Cities and Society*, Article 103828.

Pacini, E., Mateos, C., & Garino, C. G. (2014). Distributed job scheduling based on Swarm Intelligence: A survey. *Computers & Electrical Engineering*, *40*(1), 252–269.

Paidi, V., Fleyeh, H., Håkansson, J., & Nyberg, R. G. (2018). Smart parking sensors, technologies and applications for open parking lots: a review. *IET Intelligent Transport Systems*.

Shaaban, M. F., Mohamed, S., Ismail, M., Qaraqe, K. A., & Serpedin, E. (2019). Joint planning of smart EV charging stations and DGs in Eco-friendly remote hybrid microgrids. *IEEE Transactions on Smart Grid*, *10*(5), 5819–5830.

Shoup, D. C. (2006). Cruising for parking. *Transport Policy*, *13*(6), 479–486.

Shoup, D. (2017). *The high cost of free parking*. Routledge.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, *362*(6419), 1140–1144.

Singh, S., Sharma, P. K., Yoon, B., Shojafar, M., Cho, G. H., & Ra, I.-H. (2020). Convergence of blockchain and artificial intelligence in IoT network for the sustainable smart city. *Sustainable Cities and Society*, *63*, Article 102364.

Vondra, M., Becvar, Z., & Mach, P. (2016). Vehicular network-aware route selection considering communication requirements of users for ITS. *IEEE Systems Journal*, *12*(2), 1239–1250.

Wang, L., Huang, P., Wang, K., Zhang, G., Zhang, L., Aslam, N., et al. (2019). RL-based user association and resource allocation for multi-UAV enabled MEC. In *2019 15th international wireless communications & mobile computing conference* IWCMC, (pp. 741–746). IEEE.

Wang, L., Wang, K., Pan, C., Xu, W., Aslam, N., & Hanzo, L. (2020). Multi-agent deep reinforcement learning based trajectory planning for multi-UAV assisted mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking*.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3–4), 279–292.

Yue, L., Abdel-Aty, M. A., Wu, Y., & Farid, A. (2019). The practical effectiveness of advanced driver assistance systems at different roadway facilities: System limitation, adoption, and usage. *IEEE Transactions on Intelligent Transportation Systems*.

Zhong, H., Ni, J., Cui, J., Zhang, J., & Liu, L. (2021). Personalized location privacy protection based on vehicle movement regularity in vehicular networks. *IEEE Systems Journal*.

Zhu, H., Shen, L., & Ren, Y. (2022). How can smart city shape a happier life? The mechanism for developing a Happiness Driven Smart City. *Sustainable Cities and Society*, *80*, Article 103791.



https://dspace.org/

Deep reinforcement learning-based long-range autonomous valet parking for smart cities

Khalid, Muhammad; Wang, Liang; Wang, Kezhi; Aslam, Nauman; Pan, Cunhua ...

2023-02-01
Elsevier
https://doi.org/10.1016/j.scs.2022.104311
Khalid M, Wang L, Wang K, et al., (2023) Deep reinforcement learning-based long-range autonomous valet parking for smart cities. Sustainable Cities and Society, Volume 89, February 2023, Article number 104311

Downloaded from a DSpace repository. https://dspace.org/