Energy-efficient Smart Parking for Self-driving Vehicles

Xiao Chen
Department of Computer Science, Texas State University, San Marcos, TX 78666
Email: xc10@txstate.edu

***Abstract*—Self-driving vehicles have attracted tremendous attention from all walks of life and parking prediction is essential for the vehicle to find a parking space. Many existing works focus on parking predictions in an sensor network equipped environment using large datasets. In this paper, we consider a campus-wide smart parking problem that requires no sensor installation and external large datasets. We assume that the parking availabilities of the garages are not known and the only information that a vehicle uses is its past recorded statistics. We get insights from the multi-armed bandits problem in reinforcement learning and come up with a framework for a vehicle to make a wise decision in each trial. We also propose several non-reinforcement learning algorithms for comparison. We conduct extensive simulation and discussion to evaluate the performance of these algorithms.**

***Index Terms*—energy-efficient, multi-armed bandits, self-driving, smart parking**

I. Introduction

Recently self-driving vehicles have attracted tremendous attention from all walks of life. The IEEE researchers predict that self-driving vehicles will comprise 75% of total traffic on the road by year 2040 [1]. Self-driving vehicles will provide tremendous benefits to society in terms of convenience and quality of life. But the benefits will not be realized unless vehicles are truly autonomous. At this technology level, the driver hands over control to the vehicle and is no longer monitoring the system. This means that the vehicle will be able to handle all the situations on the road.

One problem facing self-driving vehicles is how to find a parking space. The parking problem is an important issue because long searching wastes time, causes more gasoline consumption and environmental pollution. Researchers have proposed prediction models to forecast parking probability. Many of these models are based on sensor-equipped environment with network connections [10], [12], [15], [17] and/or large collected datasets so that various kinds of mathematical analysis and machine learning models [3], [6], [17], [18] can be applied. These methods need hardware and software investment. Different from these works, we consider a campus-wide environment without hardware installation and expensive data collection. We are aware of two papers that also address campus-wide parking problem [2], [7]. In both papers, the authors still rely on manual data collection and an id scanner installation in the garages to make parking space predictions. In our work, we ask very little from the environment and external data collection. The only information we have is the previous simple statistics recorded by the vehicle itself.

More specifically, we address the following parking problem: we assume that a self-driving vehicle goes to a campus every day for study or work. There are several garages on the campus whose locations are known but whose parking availabilities are not known. The vehicle makes a prediction, picks a garage and drives there. If the chosen garage has an available space, the vehicle parks there. If not, the vehicle will drive to the next choice and do so until it finds a place to park. The optimization goal of our parking problem is to minimize the vehicle's total energy consumption over a number of days. Here, the energy consumption is proportional to the driving distance. So equivalently, our goal is to minimize the total driving distance of the vehicle over some number of days.

Our defined problem involves prediction and decision making with uncertainty, which is a challenge. We relate it to the multi-armed bandits problem [9] for which reinforcement learning [14] provides a model in the dilemma. Generally speaking, when facing uncertainty in decision making, reinforcement learning tries to balance exploration and exploitation. Exploration is where we gather new information that might lead to better decisions later and exploitation is where we make the best decision given the current information. Through exploration and exploitation, we plan to work on algorithms to let the vehicle make wise decisions on where to go for parking.

In this paper, we introduce and extend the eGreedy and the Upper Confidence Bound (UCB) [11] algorithms for the multi-armed bandit problem to predict the availabilities of the garages and then combine them with the driving distances to come up with *Energy-Efficient Garage Picking* (EEGP) framework to minimize vehicle energy consumption over a period of time. To evaluate the performance of our algorithms, we propose several other non-reinforcement learning algorithms. We present the *Direct* algorithm where the availabilities of the garages are known so that the vehicle can go directly there. This is the optimal case. We also include the *Random* algorithm where a vehicle randomly picks a parking garage in each try. In addition, we add an algorithm called *Closest-first* where a vehicle tries the closest garage first each time.

In summary, our major contributions in this paper are:

- We consider a realistic parking problem for self-driving vehicles on a campus with multiple garages whose availabilities are unknown, and define a smart parking prediction problem with energy constraint.
- We investigate the solutions by relating the problem to

the multi-armed bandits problem, digesting the ideas in reinforcement learning, and combining them with driving distances to form a framework for decision making.
- We also propose several non-reinforcement learning algorithms for comparison.
- We conduct extensive simulations and a discussion to evaluate the performance of the proposed algorithms.

The rest of the paper is organized as follows: Section II cites the related work; Section III defines the problem; Section IV puts forward our solutions; Section V presents the simulations and discussion; and conclusion is in Section VI.

II. Related Work

Parking problem is one of the most significant issues in self-driving vehicles. Long cruising time of finding a parking space causes enormous gasoline wastes and environmental pollution. Thus many parking prediction algorithms in smart cities have been popped up by researchers in the past few years. Many of these methods require sensor networks and/or collected large datasets on which researchers can apply multivariate regression models [3], [5], [16], Poisson distribution and Markov chain models [10], [13], [18], and other machine learning based algorithms [6], [4], [15] to provide drivers with a possibility of getting a parking space. Deep learning-based models [12], [17] are also gaining popularity.

Besides the above works, little attention has been paid to the availability prediction for the parking lots unequipped with sensors and networks. In this paper, we consider such garages, which are very common nowadays. And we do not utilize large external datasets either. We just use simple parking statistics recorded by the vehicle itself in the previous days. We investigate a practical recurring problem that many face when we drive to a campus everyday. We notice that two other papers [2], [7] also discuss the campus-wide parking problem. In [2], the authors utilized machine learning algorithms to predict future parking occupancy rates for the 34 parking lots on the campus of Charles Sturt University, Australia. They collected parking data (car occupancy and class load) manually over a five-week period. And on the campus of James Madison University [7], the researchers used a radio-frequency id scanner to count the number of vehicles entering and leaving each parking lot and developed mobile apps to inform drivers of the availability of the garages via a Cloud Environment. Different from these papers, we ask very little from the hardware installation and external dataset collection in our work.

III. Problem Definition

In our problem, there are a set of parking garages on a campus without sensors and networks installed to monitor the parking spaces. On a typical workday, a self-driving vehicle is trying to park in one of the garages. The parking availability of each garage is not known and the vehicle only relies on its own past recorded statistics and driving distance to pick a garage and drives there. If the garage has a space, the vehicle is parked. Otherwise, the vehicle makes the next choice. It repeats this process until it finds a parking space. We assume that eventually the vehicle is able to find a space to park. Our optimization goal is to minimize the total driving distance of the vehicle over some number of days so that the total energy consumption can be minimized. That means, in each parking trial, the vehicle needs to balance its garage availability prediction and the driving distance so that the total driving distance over a period of time can be minimized.

Notation	Description
cur_loc	The current location of the vehicle
$dist(X, Y)$	The distance between locations X and Y
$reward$	If a garage has a space, the reward is 1; 0 otherwise
$rewards$	An array containing the rewards of all the garages
$rewards(g)$	The rewards of garage g
$score$	An array containing the scores of all the garages
$score(g)$	The score of garage g
$trials$	An array containing the trial times of all the garages
$trials(g)$	The trial times of garage g
$total\_distance$	The total distance the vehicle drives before parking

TABLE I
Notations

IV. Our Solutions

In this section, we investigate the solutions to our defined problem. First we digest the ideas in reinforcement learning and propose an Energy-efficient Garage Picking (EEGP) framework for the vehicle to choose a garage. Then we propose several non-reinforcement learning algorithms for later comparison. The notations used in the framework are listed in Table I.

A. Energy-Efficient Garage Picking (EEGP) Framework

In the EEGP framework in Fig. 1, we relate the garage availability prediction part of our problem to the multi-armed bandits problem. Here, the garages on campus are the arms. The main idea of the framework is as follows: as long as the vehicle has not parked, it is going to repeat the following steps. First it will call X_pick() function (line 3) to choose the best untried garage after balancing exploration and exploitation, the current location of the vehicle, and the driving distance consideration. X_pick() is a placeholder. It has two versions eGreedy_pick() and UCB_pick() based on the ideas of the two multi-armed bandit algorithms eGreedy and UCB [11], respectively. Different versions take different parameters, but they have $score, cur\_loc, dist$ in common. The details of the two versions will be described in the following subsections. After a garage g is chosen, the vehicle will drive there and the driving distance from the current location to the garage $dist(cur\_loc, g)$ is added to the $total\_distance$ (line 5) driven by the vehicle. If the reward of g is 1, which means the garage has a parking space, the vehicle is parked (line 6). After that, function updateScore() will be called to update the garage information based on this trial (line 10).

The details of updateScore() are presented in Fig. 2. In this function, the trial times of the picked garage g is incremented by 1 because it is selected for this trial. The rewards of g is incremented by the $reward$ in this trial. The value of $reward$ is 1 if g has an available spot, 0 otherwise. The score of g is the ratio of $rewards(g)$ and $trials(g)$.

Algorithm EEGP: Energy Efficient Garage Picking

Require: Input: $score$, cur_loc, $dist$, $rewards$, $trials$
Output: a garage is picked and the vehicle parks there

1: **while** not parked **do**
2: /*X_pick() is to select an untried garage based on the exploited garage information, the current vehicle location, and the distance information */
3: g = X_pick(score, current_location, distances, $\cdots$);
4: /* the distance from the current vehicle location to the picked garage g is added to the total driving distance */
5: $total\_distance = total\_distance + dist(cur\_loc, g)$;
6: **if** the reward of g is 1 (space available) **then**
7: parked = true;
8: **end if**
9: /*updateScore() is to update the garage information based on this trial */
10: updateScore(trials, rewards, score, g, reward);
11: **end while**

Fig. 1. Energy Efficient Garage Picking (EEGP) Framework

Function: updateScore()

Require: Input: $trials$, $rewards$, $score$, g, $reward$
Output: updated $trials$, $rewards$, $score$

1: /* Assume the picked garage is g */
2: /* add 1 to the trial times of the picked garage g */
3: $trials(g) = trials(g) + 1$;
4: /* add reward 1 to the rewards of the picked garage g */
5: $rewards(g) = rewards(g) + reward$;
6: /* update the score of the picked garage g */
7: $score(g) = rewards(g)/trials(g)$;

Fig. 2. updateScore() function

1) EEGP with eGreedy Picking: In this subsection, we describe the idea of the eGreedy version of the X_pick() function eGreedy_pick() shown in Fig. 3. In eGreedy_pick(), the tradeoff between the exploration and exploitation is controlled by a parameter ϵ (line 1). If the number $rand$ generated by a random number generator is less or equal to ϵ, the vehicle explores the availabilities of the garages by randomly picking a garage (line 2). Otherwise it does exploitation to make the best decision by considering the scores of the untried garages and the distances from the current vehicle location to the untried garages (line 3). The value of ϵ is set to a number between 0 and 1, say 0.1. It is usually small so we mainly make a decision based on exploitation but still leave a little room to explore the unknown availabilities of the unexplored garages. In the exploitation, we normalize $score(g)$ and $dist(cur\_loc, g)$ to the same scale (line 5) and then pick a garage with the largest score/distance ratio.

2) EEGP with UCB Picking: The UCB version of the X_pick() function is described in Fig. 4. It uses a different idea. The true availability score denoted by $SCORE(g)$ of each garage g is not known. The $score(g)$ we obtain from the exploration trials is the sample score. Through the trials, we want $score(g)$ to be as close to $SCORE(g)$ as possible,

Function: eGreedy_pick()

Require: Input: ϵ, $score$, cur_loc, $dist$
Output: The garage picked in this trial

1: **if** $rand <= \epsilon$ **then**
2: Randomly pick a garage
3: **else**
4: **for** g in all untried garages **do**
5: normalize $score(g)$ and $dist(cur\_loc, g)$;
6: **end for**
7: pick a garage with the largest $score(g)/dist(cur\_loc, g)$ ratio;
8: **end if**

Fig. 3. eGreedy_pick() function

or at least probabilistically. To do that, we have Hoeffding's inequality [8]:

$$Pr(SCORE(g)) \leq score(g) + \sqrt{\frac{\log(\frac{1}{\delta})}{2N_t(g)}}) \geq 1 - \delta \quad (1)$$

Hoeffding's inequality shows that the true score $SCORE(g)$ is upper bounded by the sample score $score(g)$ plus $\sqrt{\frac{\log(\frac{1}{\delta})}{2N_t(g)}}$. Here, the parameter δ is a value in $[0, 1]$. Notation $N_t(g)$ means that in t trials, the number of times that the vehicle has chosen garage g. With a certain δ, if the vehicle chooses garage g more often in all the t trials, the value $\sqrt{\frac{\log(\frac{1}{\delta})}{2N_t(g)}}$ will decrease and therefore the upper bound becomes tighter, which means that we are closer to discover the true score of g. For δ, if we make it smaller and smaller with the trial times t, then the probability to discover the true score will become larger and larger. In the UCB algorithm [11], $\delta = \frac{1}{t^4}$. Then Hoeffding's inequality (1) becomes

$$Pr(SCORE(g)) \leq score(g) + \sqrt{\frac{2 * \log(t)}{N_t(g)}}) \geq 1 - \frac{1}{t^4} \quad (2)$$

This is the reasoning behind the formula in line 3 of the UCB_pick() function in Fig. 4. In the UCB_pick() function, initially each garage will be tried once to obtain its initial score. Then for all the untried garages, we update their $score(g)$ values using the formula in line 3 and then normalize them with the distances from the current vehicle location to these garages. Finally we pick the garage with the largest score/distance ratio.

B. Random Algorithm

In the Random algorithm, the car randomly chooses a garage to park in each try until it finds a parking space.

C. Direct Algorithm

In this algorithm, we assume that the vehicle knows the closest garage that has an available spot and drives there directly. This is an ideal but impractical strategy. But it serves as the best case benchmark in the algorithm comparison.

D. Closest-First Algorithm

In this algorithm, the vehicle always drives to the closest garage first based on its current location.

Function: UCB_pick()

Require: Input: $score$, cur_loc, $dist$
Output: The garage picked in this trial

1: **for** g in all untried garages **do**
2: /* $score(g)$ is the score of garage g so far, t is the total number of trials, and $N_t(g)$ is the number of times garage g has been picked in the t trials */
3: $score(g) = score(g) + \sqrt{\frac{2*\log(t)}{N_t(g)}}$;
4: **end for**
5: **for** g in all untried garages **do**
6: normalize $score(g)$ and $dist(cur\_loc, g)$;
7: **end for**
8: pick a garage with the largest $score(g)/dist(cur\_loc, g)$ ratio;

Fig. 4. UCB_pick() function

V. Simulations and Discussion

In this section, we compare the performance of our proposed algorithms using a simulator written in Matlab and discuss the applicability of the algorithms.

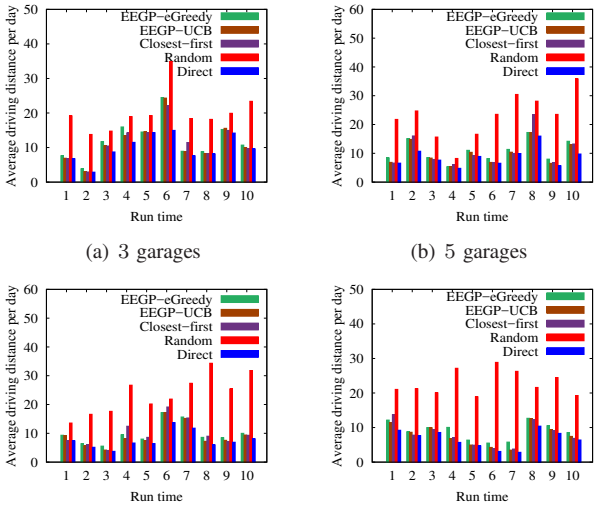
A. Comparison of All the Algorithms

In this simulation, we compare the following algorithms.

1) *The EEGP with eGreedy Algorithm* (EEGP-eGreedy)
2) *The EEGP with UCB Algorithm* (EEGP-UCB)
3) *The Random Algorithm* (Random)
4) *The Direct Algorithm* (Direct)
5) *The Closest-first Algorithm* (Closest-first)

We assume that the parking availabilities of the garages are stochastic. We tried 3, 5, 8, and 10 garages with randomly generated locations on a campus. We randomly generated the availabilities of these garages in the range of $[0, 1]$. In the EEGP-eGreedy algorithm, we set ϵ to 0.1. We looked at 300 working days in a year on each of which a vehicle wants to find a place to park. For all the algorithms, we calculated the average distance to find a parking space per day over the 300 working days. The results of 3, 5, 8, and 10 garages are shown in Figs. 5(a), (b), (c), and (d), respectively. The numbers in the x-coordinate represent the 10 times that we ran the simulations. The y-coordinate is the average driving distance per day.

From the figures we can see that the Random algorithm produces the longest average driving distance per day and the Direct algorithm has the shortest driving distance per day. This is because the Random algorithm does not use any guidance in finding a parking space while the Direct algorithm knows exactly which garage is available and can go directly there. They provide the upper and lower bounds of the average driving distance. The EEGP-eGreedy algorithm has longer average distance than the EEGP-UCB algorithm. This is because the random exploration in EEGP-eGreedy may end up with a bad choice but the exploration in EEGP-UCB favors choices with a strong potential to be the best choice. As for the Closest-first and the EEGP-UCB algorithms, sometimes Closest-first performs better and sometimes EEGP-UCB performs better. In the following simulations, we will compare them more in detail to find out in what situation is one better than the other.



(a) 3 garages (b) 5 garages

(c) 8 garages (d) 10 garages

Fig. 5. Average driving distance per day to find a parking space

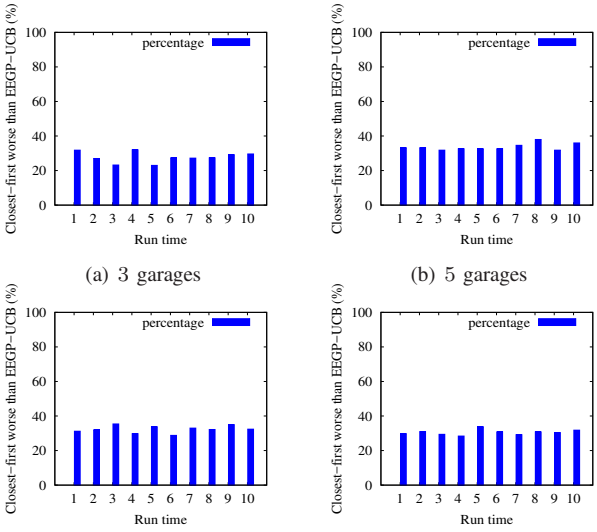
B. Availability Randomly Distributed

In this simulation, we compare the performance of the Closest-first algorithm and the EEGP-UCB algorithm. We assume that the availabilities of the garages are random. For the 3, 5, 8, and 10 garages, we randomly generated the availabilities of these garages in the range of $[0, 1]$. We set the number of days we observe to 300. For the two algorithms, we first calculated the average driving distance to find a parking space per day over the 300 working days and then ran the simulation 1000 times to find out the percentage of the times that the Closest-first algorithm has a longer average driving distance per day than the EEGP-UCB algorithm. The percentage results of the 3, 5, 8, and 10 garages are shown in Figs. 6(a), (b), (c) and (d), respectively. The numbers in the x-coordinate represent the 10 times that we ran the 1000-time simulations and the y-coordinate is the percentage that the Closest-first algorithm is worse than the EEGP-UCB algorithm.

From the figures we can see consistently that the chance of the Closest-first algorithm having a longer average driving distance per day than the EEGP-UCB algorithm is below 50%. So we can conclude that if the availability is randomly distributed, it is better to use the Closest-first algorithm.

C. Availability Not Randomly Distributed

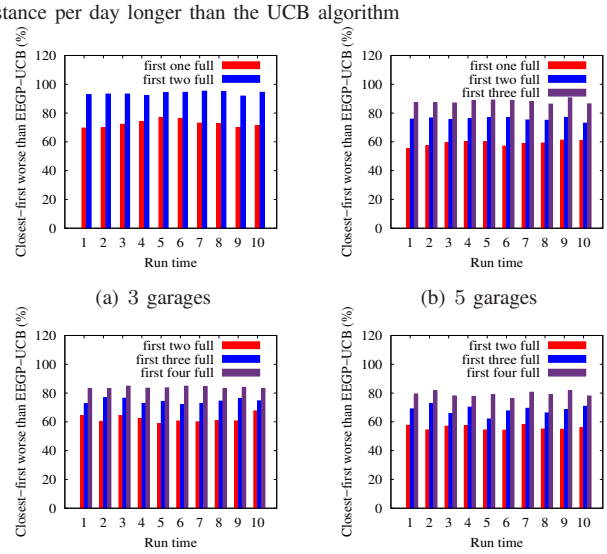
In this simulation, we compare the performance of the Closest-first algorithm and the EEGP-UCB algorithm by assuming that the availabilities of the garages are not random. We still tried 3, 5, 8, and 10 garages. We assume the availabilities of the garages are increasing with the increase of the distances. This is a reasonable assumption because usually the closer garages are filled first. We set the first 20% or more garages full and randomly generated the availabilities of the rest of the garages in an increasing order in the range of $[0, 1]$. If the



(a) 3 garages (b) 5 garages

(c) 8 garages (d) 10 garages

Fig. 6. Percentage of the times that the closest-algorithm has an average distance per day longer than the UCB algorithm



(a) 3 garages (b) 5 garages

(c) 8 garages (d) 10 garages

Fig. 7. Percentage of the times that the closest-algorithm has an average distance per day longer than the UCB algorithm

number of the garages was not an integer, we rounded it up to the ceiling. We again set the number of days to 300. For the two algorithms, we first calculated the average driving distance to find a parking space per day over the 300 working days and then ran the simulation 1000 times to find out the percentage of the times that the Closest-first algorithm has a longer average driving distance per day than the EEGP-UCB algorithm. The percentage results of the 3, 5, 8, and 10 garages are shown in Figs. 7(a), (b), (c), and (d), respectively. The numbers in the x-coordinate represent the 10 times that we ran the 1000-time simulations and the y-coordinate is the percentage that the Closest-first algorithm is worse than the EEGP-UCB algorithm.

From the figures, we can see that when the first few closest garages (over 20% of the garages in the simulations) are full, above 50% of the chance that EEGP-UCB will have a shorter average driving distance per day than Closest-first. Therefore, we can conclude that in this situation, it is wiser to use the EEGP-UCB algorithm to find the parking space.

VI. Conclusion

In this paper, we have worked on a parking problem with energy constraint on a campus where the availabilities of the garages are unknown for self-driving vehicles. We have investigated the solutions by digesting the ideas in reinforcement learning and combining them with driving distances to form a framework for picking the best garage in each trial. We have also proposed some non-reinforcement learning algorithms. We have conducted extensive simulations and discussion to evaluate the performance of the algorithms. In the future, in order to improve the prediction accuracy, we will take more factors into account such as time of day and holiday.

References

[1] http://www.ieee.org/about/news/2012/5september_2_2012.html.

[2] J. A. Barker and S. Rehman. Investigating the use of machine learning for smart parking applications. *11th International Conference on Knowledge and Systems Engineering (KSE)*, pages 1–5, 2019.

[3] S. S. Chawathe. Using historical data to predict parking occupancy. *IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*, pages 0534–0540, 2019.

[4] X. Chen. Parking Occupancy Prediction and Pattern Analysis. http://cs229.stanford.edu/proj2014/XiaoChen,ParkingOccupancyPredictionand, PatternAnalysis.pdf.

[5] J. Fan, Q. Hu, and Z. Tang. Predicting vacant parking space availability: an svr method with fruit fly optimisation. *IET Intelligent Transport Systems*, 12(10):1414–1420, 2018.

[6] X. Fang, R. Xiang, L. Peng, H. Li, and Y. Sun. Saw: A hybrid prediction model for parking occupancy under the environment of lacking real-time data. *Annual Conference of the IEEE Industrial Electronics Society*, pages 3134–3137, 2018.

[7] M. Garcia, P. Rose, R. Sung, and S. El-Tawab. Secure smart parking at james madison university via the cloud environment (space). *IEEE Systems and Information Engineering Design Symposium*, 2016.

[8] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[9] M. N. Katehakis and A. F. Veinott. The multi-armed bandit problem: Decomposition and computation. *Mathematics of Operations Research*, pages 262–268, 1987.

[10] A. Klappenecker, H. Lee, and J. L. Welch. Finding available parking spaces made easy. *Ad Hoc Networks*, 12:243–249, 2014.

[11] T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[12] Jiachang Li, Jiming Li, and Haitao Zhang. Deep learning based parking prediction on cloud platform. *BIGCOM*, pages 132–137, 2018.

[13] L. Peng and H. Li. Searching parking spaces in urban environments based on non-stationary poisson process analysis. *IEEE ITSC*, pages 1951–1956, 2016.

[14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[15] E. I. Vlahogianni, K. Kepaptsoglou, V. Tsetsos, and M. G. Karlaftis. A Real-Time Parking Prediction System for Smart Cities. *Journal of Intelligent Transportation Systems*, 20(2):192–204, 2016.

[16] Y. Wu. Analysis of present status and prediction of future demand on parking in the central area of medium and small city. *International Conference on Remote Sensing, Environment and Transportation Engineering*, pages 57–60, 2011.

[17] S. Yang, W. Ma, X. Pi, and S. Qian. A deep learning approach to real-time parking occupancy prediction in transportation networks incorporating multiple spatio-temporal data sources. *Transporation Research Part C: Emerging Technologies*, 107:248–265, 2019.

[18] L. Zheng, X. Xiao, B. Sun, D. Mei, and B. Peng. Short-term parking demand prediction method based on variable prediction interval. *IEEE Access*, 2020.