A reinforcement learning approach for user-optimal parking searching strategy on a network exploiting network topology

Jun Xiao, Ph.D. Student
School of Sustainable Engineering and Built Environment
Arizona State University
660 S. College Avenue, Tempe Arizona 85281
Tel: 480-274-5418, Email: jun.xiao.1@asu.edu

Yingyan Lou, Ph.D., Corresponding Author
School of Sustainable Engineering and Built Environment, Arizona State University
660 S. College Avenue, Tempe Arizona 85281
Tel: 480-965-6361, Email: yingyan.lou@asu.edu

Submitted
to
2019 Transportation Research Board Annual Meeting
for
Publication and Presentation

August 1, 2018

Word Count:

Cover page:	*100*
Abstract:	*204*
Manuscript:	*4615*
Tables:	*2 (equivalent to 500 words)*
Reference:	*943*

Total: 6362

Abstract

This paper investigates the idea of introducing learning algorithms into parking guidance and information systems that employ a central server, in order to provide estimated optimal parking searching strategies to travelers. The parking searching process on a network with uncertain parking availability can naturally be modeled as a Markov Decision Process (MDP). Such an MDP with full information can easily be solved by dynamic programming approaches. However, the probabilities of finding parking are difficult to define and calculate, even with accurate occupancy data. Learning algorithms are suitable for addressing this issue. The central server collects data from numerous travelers' parking search experiences in the same area within a time window, computes approximated optimal parking searching strategy using a learning algorithm, and distributes the strategy to travelers. We propose an algorithm based on Q-learning, where the topology of the underlying transportation network is incorporated. This modification allows us to reduce the size of the problem dramatically, and thus the amount of data required to learn the optimal strategy. Numerical experiments conducted on a toy network show that the proposed learning algorithm outperforms the nearest-node greedy search strategy and the original Q-learning algorithm. Sensitivity analysis regarding the desired amount of training data is also performed.

1. Introduction

As an essential component of the urban transportation network, parking searching has been widely considered as a significant reason for congestion in downtown areas. According to empirical studies (*1, 2*), cruising for parking is responsible for 30% of traffic (trips) on average in urban areas. In a more recent research, Giuffrè et al. (*3*) found that cruising for parking results in a peak increase of about 25 – 40% of the traffic flow. Simulation studies incorporating parking search at macroscopic (*4–6*) and microscopic level (*7*, *8*) also show the degradation of traffic condition caused by cursing for parking. For example, by exploiting the properties of the macroscopic fundamental diagram, Geroliminis (*5*) modeled the dynamics of parking searching and showed the cruising affects all travelers even those with a destination outside the limited parking region.

Recognizing the significance of parking searching problem, parking guidance and information systems, which often involve parking information collection, processing, and distribution, have emerged to help drivers find parking spaces. Parking information collection is often achieved by sensors at entrances/exits or at individual parking stalls (*9*). Crowdsourcing is also an emerging approach to parking information collection. Furthermore, to provide future parking availability information, many studies have proposed prediction methods with model-based, statistical, or machine learning approaches (e.g., *10*, *11,* and the articles cited therein). Data distribution relies on efficient communications between drivers and the service provider (*12*, *13*); and smartphone-based services have become a popular solution. Available mobile services currently on the market include mobile payment, parking information provision, and reservation (e.g., ParkMobile, ParkMe, BestParking etc.).

We argue, however, the effectiveness of providing parking information (current parking availability, predicted future parking availability, and predicted parking search time) alone to users might be limited. The parking information collection methods mentioned previously may be unreliable under abnormally high parking demand, which usually suggests special events and unfamiliar drivers, when effective parking guidance and information services are most needed. Under such circumstances, it is common that parking spaces are reassigned or reserved (special event); and double parking is also common for parking facilities with narrow stalls (unfamiliar drivers). While some researchers found parking information such as occupancy information, parking searching time (*14*) and future parking availability (*15*) significantly reduce users' cruising time, others showed that the benefit depends on the level of congestion (*16*). The benefit is limited when the parking capacity is almost saturated.

Providing travelers with parking searching strategies (suggested actions for a user to take in any given state of the parking searching process) could be a more effective approach to parking management, especially for unfamiliar drivers and for special events. The parking searching problem can naturally be viewed as a Markov Decision Process (MDP) where the outcome is random. User-optimal strategies can be formed if the underlying Markov transition probabilities and the cost/reward structure associated with the outcomes are known. To make the MDP formulation more realistic, Tang et al. (*17*) introduced driver memories in modeling individual driver's parking searching process. Drivers are assumed to have memories of the probability of finding parking at each facility, which is set to either 1 or 0 upon the outcome of a visit to a facility, and are later gradually reset to an *a priori* value. In their approach, the state space would grow exponentially with the memory size. Additionally, the actual probabilities of

finding parking (the *a priori* values) are difficult to define and calculate, even with accurate occupancy data. To address this issue, learning algorithms can be introduced to approximate the optimal strategy. Commuters already form user-optimal strategies by learning from their day-to-day experiences. While learning is limited during non-recurring parking searching for an individual driver (e.g., when going to a special event or visiting a special destination) due to limited interaction with the environment, it is possible to introduce learning into parking guidance and information services that can leverage experiences from numerous users attending the same event or visiting the same destination.

In this paper, we consider the setting where a central server is employed by a parking guidance and information service provider, and users of the service interact with the central server through a mobile phone app or alike. Figure 1 describes the users' interactions with the server and the environment. Through the app, the central server gathers information on whether a user successfully finds parking or not either through location data or user reports. It then learns from the parking search experiences from all the users in the same area within a reasonable time window to compute an approximated user-optimal strategy. The approximated user-optimal strategy is distributed to those still searching for parking within the same time window through the mobile app. This learning cycle repeats for each time window. For users whose searches are not completed within a time window / learning cycle, updated recommendations can be pushed to them through the mobile app. Alternatively, this update can be skipped and the outcomes of these users' searches following old strategies would provide valuable data and jump-start the next learning cycle. The latter might even be preferred by the users, especially when the time windows are small, so that they do not receive constant updates.

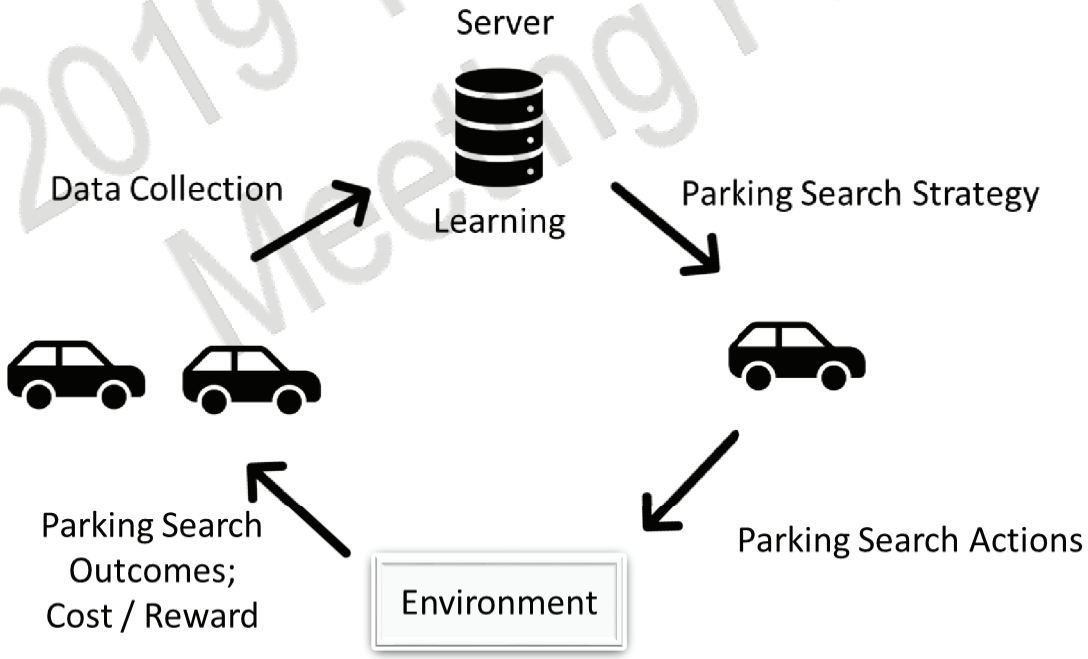

FIGURE 1 FRAMEWORK

Note that the users are not required to adopt the recommended strategy. Diverse actual behavior policies adopted by the users arguably would help with the convergence of the learning

algorithm. While travelers' actual searching behaviors would affect the underlying probabilities of finding parking at each facility, it should also be noted that we are not trying to predict the actual parking probabilities for the current and future time windows. The actual probabilities are highly likely to fluctuate continuously; but the proposed system and learning algorithm will model the probabilities as constant in a given time window / learning cycle.

Therefore, it is desirable that the parking conditions are relatively stable during each time window / learning cycle (from drivers exploring the environment and collecting data to an estimated user-optimal strategy being distributed to drivers) as shown in Figure 1. Otherwise, the knowledge and user-optimal strategy learned during exploration would be applied in a different environment and would not be effective. The time windows could be pre-determined for recurrent special events (such as seasonal sport events) where the demand patterns are generally known. They could also be determined in real time when the central server detects a shift in the parking search outcomes from the users; or if the parking guidance and information service provider is also able to obtain data (such as real-time occupancy and capacity of a parking facility) from infrastructure managers. With data from the supply side, it is possible to estimate the parking probabilities and predict future parking availability. Our previous work (*11*) has proposed a rolling-horizon framework and an iterative approach for this purpose. The framework is also able to detect demand pattern shifts, and could potentially be applied for the time window determination.

The focus of this paper, on the other hand, is on reducing the amount of data and time required to train the learning algorithm. This is crucial so that the time window can be sufficiently short, if needed. Additionally, being able to learn the optimal strategy with limited data frees the parking guidance and information service from a high market penetration requirement. The service and algorithm would not need all or a large percentage of the travelers to participate in order to work.

We propose a modified Q-learning algorithm that can be adopted by such parking guidance and information services to provide user-optimal parking search strategies for individual drivers. As a reinforcement learning algorithm, Q-learning is a model-free off-policy method where the learning is performed through agent actions and environment feedback. Since user actions are choosing the parking facility to visit next, and the reward structure is closely coupled with the transportation network through travel cost, it is possible to incorporate and take advantage of the network topology in the learning algorithm. We propose a Q-learning-based algorithm that utilizes the underlying transportation network topology, which can dramatically reduce the size of the state space and improves the convergence speed and the solution quality.

To this end, the contributions of this paper are: 1) the first to introduce reinforcement learning into parking guidance and information systems to provide user-optimal parking strategies to individuals; 2) a modified reinforcement learning algorithm utilizing the topology of the transportation network, greatly reducing the amount of data required for training.

2. Methodology

This section discusses the model and methodology proposed in this paper. Section 2.1 describes our model for the parking searching MDP problem. Section 2.2 provides an overview on reinforcement learning methods. Section 2.3 explains in detail the proposed learning algorithm that utilizes the transportation network topology.

2.1 Modeling parking searching problem as a Markov decision process

Consider a strongly connected urban transportation network with a set of parking facilities (nodes), denoted as $N = \{1, 2, \dots, n\}$, where there is a path between every pair of nodes. The probability of finding parking at each facility i is denoted as p_i. This network model can incorporate both off- and on-street parking facilities. For on-street parking, a dummy node can be created with its corresponding probability and inserted in the middle of the modeled street. A driver starts from origin O, tries to find parking in the transportation network, and travels to destination D using other modes (e.g., walking). Without loss of generality, we assume that no parking is allowed at the origin O or the destination D. The goal of a parking strategy is to minimize a driver's expected cost (or maximize a driver's reward) of parking, which includes the cost of cruising to search for parking and the travel cost from a parking facility to her final destination. For the non-recurring parking searching scenarios considered in this paper (e.g., when going to a special event or visiting a special destination), p_i is unknown to the users or the parking guidance and information system. While p_i constantly fluctuates in reality, we assume that p_i remains relatively stable in each time window / learning cycle as discussed in Section 1; and the learning algorithm will treat p_i as constant.

The parking searching problem can be formulated as an MDP. Each node i in the set N corresponds to two states: i_R and i_P, representing a traveler not finding parking / found parking at node i respectively. If a traveler is in state i_R, she can take action a_{ij} $(i, j \in N, j \neq i)$, representing the action of driving from node i to node j and searching for parking there. Note that node j may not be the immediate adjacent facility to node i in the physical road network. Each action a_{ij} leads to two possible outcomes (j_R and j_P) depending on the probability of finding parking at node j. If a traveler is in state i_P, the only action allowed next is a_{iD}, traveling to the destination using other modes.

Figure 2 and Figure 3 illustrate the physical network and the transition graph for a 3-node example. In Figure 2, only driving links but no walking paths are shown. Note that the origin node is not directly connected to nodes 2 and 3 (Figure 2), but a driver at origin O can take actions $a_{oj}, \forall j \in N$ (Figure 3). Action a_{02} means that the driver goes from origin to node 2 to search for parking, bypassing node 1 even it is on his way. A driver in state $the\ 1_R$ (physically at node 1 but did not find parking there) can take action a_{1j} $(j = 2, 3)$, driving to node j and searching for parking there. If he takes action a_{12}, then there is a probability of p_2 that he finds parking and reaches state 2_P; he will then take action a_{2D} to travel to the final destination using other modes (e.g., walking). Otherwise he would end up with state 2_R and continue searching. Also note that states $1_P, 2_P, \dots n_P$ are in fact terminal states, as the only action allowed from these states is traveling to the destination D using other modes. Given the states and actions above, we are able to define the rewards based on the actions taken. For searching actions, the reward of $a_{ij}(i \in N \cup \{O\}, j \in N, j \neq i)$, is simply defined as $-d(i, j)$, the negative shortest path travel distance between node i and j. Similarly, after a traveler finds parking, the reward is defined as the negative of the travel cost to the destination by other modes.

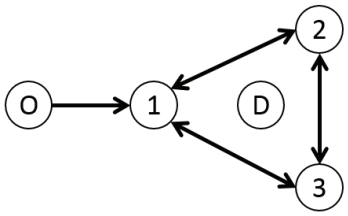

FIGURE 2 PHYSICAL NETWORK (3-NODE EXAMPLE)

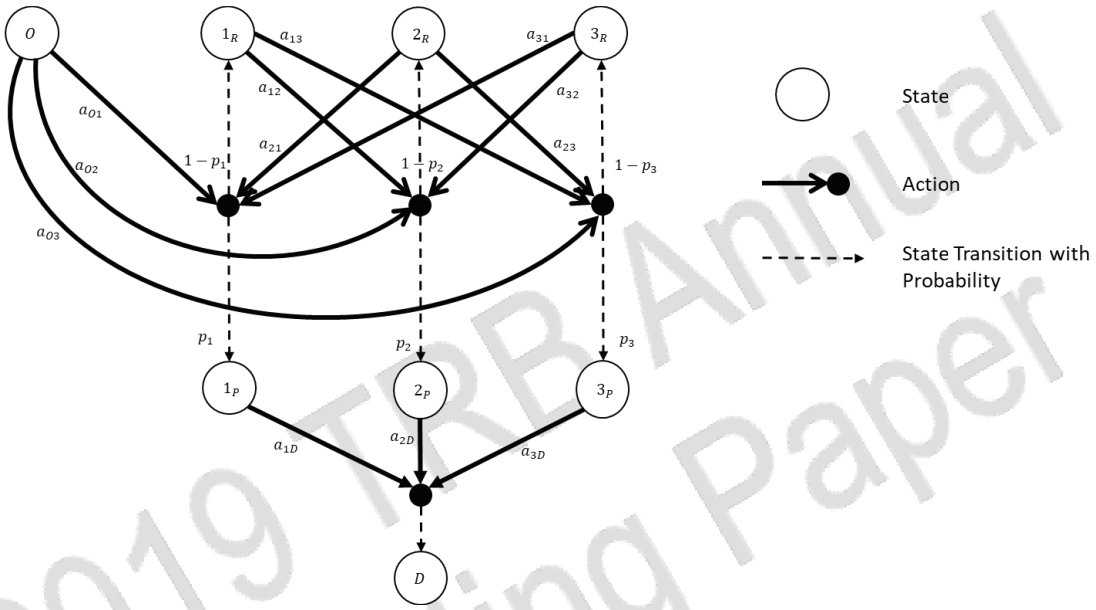

FIGURE 3 TRANSITION GRAPH (3-NODE EXAMPLE)

The optimal value of a state is the total expected reward staring from this state to the destination state D, following the optimal strategy. If the transition probabilities are known, the problem can be solved exactly using dynamic programming methods such as value iteration and policy iteration. When the transition probabilities are unknown to agents in the system, reinforcement learning methods can be adopted to estimate the optimal values and generate approximated optimal strategy based on the values learned. Note that the optimal values of terminal states $s \in \{D, 1_P, 2_P, \dots, n_P\}$ are determined. We are interested in estimating the optimal values of state $s \in \{O, 1_R, 2_R, \dots, n_R\}$.

2.2. Overview of reinforcement learning

With the advancements in theory, algorithms and computational power, reinforcement learning algorithms have already been effectively applied to transportation problems including signal control (*18–21*), train rescheduling (*22*), travel behavior (*23*) and autonomous vehicle control (*24*, *25*).

In reinforcement learning, an agent learns how to achieve a certain goal by exploring the environment: it takes an action at its current state in the environment, moves to the next state depending on the probabilistic outcome, and receives rewards. The action and reward pairs are used to update the estimated optimal value of each state.

On-policy and off-policy methods are two classes of learning methods. On-policy methods (e.g., SARSA and TD(λ), see (*27*)) start with a given policy that adopts the current optimal actions most of the time but also has small probabilities of taking other actions for further exploration, and learn the value of each state under this policy until the optimal values and actions converge. On the other hand, off-policy methods (e.g., Q-learning (*28*), R-learning (*29*)) can learn the state-action values of a target policy that the agents do not necessarily follow in their exploration.

For our problem, off-policy methods are more appropriate since the central server does not know the actual behavior policies being adopted by travelers in the parking search data collected. Q-learning is a common off-policy algorithm learning directly from the consequence of actions. It can be proven that given sufficient training data under any soft behavior policy, where the probability of taking the current optimal action is less than 1, even purely random behavior policy, the optimal action values learned will converge with probability 1 (*27*).

2.3 Proposed learning algorithm

We propose a modified Q-learning algorithm which takes advantage of the network topology and dramatically reduces the size of state space. The proposed algorithm takes travelers' parking search actions and outcomes (regardless of the actual behavior policies adopted) and generates an estimated optimal parking searching strategy. It does not require a large amount of data for training, and is able to converge quickly. It is suitable to provide an approximated optimal strategy for parking searching problem with limited information.

We will briefly introduce the Q-learning algorithm first, followed by the proposed algorithm which is based on Q-learning.

Q-learning is an off-policy learning approach approximating the underlying value of state-action function $Q(s,a)$ ($s \in S, a \in A_s$) directly when an agent at state s takes action a and ends up at state s':

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r_a + \gamma \max_{a' \in A_{s'}} Q(s',a')) \quad (1)$$

where α is the learning rate and γ is the discount factor. Since we are trying to minimize the total cost of the parking search process without discounting the cost of any action in our problem, the discount factor $\gamma = 1$. r_a is the immediate reward of action a and $\max_{a' \in A_{s'}} Q(s',a')$ is the maximal possible reward afterwards. Equation (1) means that $Q(s,a)$ would be updated as the weighted average of the current value of $Q(s,a)$ and the reward following the current optimal strategy based on current Q values. Note that the Q values can be arbitrarily initialized because in each update, intuitively speaking, we are adding more truth and discounting the effect of initialized value by α. In fact, it can be proven that the algorithm converges as long as all state-action pairs continue to be updated (*27*).

We propose a modified Q-learning algorithm for the parking searching problem, taking advantage of the topology of the transportation network. Note that $Q(i_R, a_{ij})$ represents the expected value of taking action a_{ij} from state i_R. This means that $Q(i_R, a_{ij})$ is comprised of two components: the travel cost from current node i to j, and the sum of all the expected following rewards starting from searching at node j. Now consider two state-action pairs (i_R, a_{ik}) and (j_R, a_{jk}) where the actions are driving to the same node k from different current nodes i and j. The only difference between $Q(i_R, a_{ik})$ and $Q(j_R, a_{jk})$ lies in the first component, their travel

costs. The sum of all the expected following rewards starting from searching at node k would be the same, denoted as $V(k_R)$. Define

$$V(k_R) = Q(O, a_{Ok}) - d(O,k) \\ = Q(i_R, a_{ik}) - d(i,k) = Q\left(j_R, a_{jk}\right) - d(j,k), \quad \forall i,j \in N \tag{2}$$

Since the travel cost is deterministic, we only need to update one value $V(k_R)$ for all the actions going to the same location k when any action a_{ik} is taken. From equation (2), updating $V(k_R)$ is essentially updating $Q(i_R, a_{ik}),\ \forall i \in N$ as well as $Q(O, a_{ik})$. In other words, with the original Q-learning equation (1), only one Q value is updated when a state-action pair occurs; but with equation (2), when a state-action pair occurs, the Q values for all state-action pairs with the same destination node are updated at the same time. This will greatly reduce the number of updates to be performed in the learning process.

Pseudo code:

Initialize $V(s)$, $count(s)$
Repeat (for each parking search trajectory):
  Initialize the starting node i,
  Repeat (for each parking facility j visited in the search trajectory
  before finally finding parking):
    $\alpha(j) \leftarrow \frac{1}{count(j)^{\frac{2}{3}}}$
    $V(j_R) \leftarrow (1-\alpha)V(j_R) + \alpha[r_{a_{ij}} + \underset{k \in A_{j_R}}{max}\left(V(k_R) + d(j,k)\right) - d(i,j)]$
    count(j) ← count(j) + 1
    $i \leftarrow j$

where $count(j)$ records how many times facility j is visited. The learning rate α is calculated separately for each node j based on $count(j)$. For each node, the series of learning rate must add up to infinity while the summation of square of α must be finite to ensure convergence (*31*). After estimating V values, the Q values can be derived by reversing equation (2). The optimal policy can then be derived by choosing the action with maximal value at each state.

2.4 Benchmark algorithm

To evaluate the proposed algorithm later in our numerical experiments, we choose the value iteration method as a benchmark where all the parking probabilities are assumed known. Denote $V(s)$ as the expected total rewards if a traveler starts from state s and taking optimal actions until he reaches the destination. Value iteration iteratively updates $V(s)$ by taking the optimal action based on the current $V(s)$ estimates:

$$V_{i+1}(s) = \underset{a \in A_s}{max}\{\Sigma_{s'} P_a(s,s')(R_a(s,s') + \gamma V_i(s'))\}$$

In the above equation, $V_i(s)$ is the i^{th} estimate of $V(s)$, $P_a(s,s')$ is the probability of transitioning from state s to s' under action a, and $R_a(s,s')$ is the corresponding reward. $V_0(s), \forall s \in S$ can be arbitrarily initialized. The discount factor γ is 1 because we are considering the total undiscounted reward. As an undiscounted Markov decision problem, the convergence has not been completely understood; but for our problem, the convergence of value iteration can be proved from a sufficient condition proposed in (*30*). This guarantees the validity

of using value iteration as our benchmark. Finally, the optimal state values $V(s)$ can be used to derive an optimal strategy π by:

$$\pi(s) = \arg\max_{a \in A_s} \{\Sigma_{s'} P_a(s,s')(R_a(s,s') + \gamma V_i(s'))\}$$

3. Numerical Experiments

Numerical experiments are conducted with randomly generated parking probabilities on a toy network to investigate the performance of the proposed method. The transportation network considered has six nodes. The network topology and associated travel costs as shown in Figure 4. The proposed method is compared with the original Q-learning, a greedy nearest-node strategy and the optimal strategy calculated using value iteration from the exact probabilities of finding parking. The experiments show that the proposed learning algorithm could generate a searching strategy close enough to the optimal strategy with a reasonable size of training data. Additionally, sensitivity analysis is performed regarding the amount of data needed.

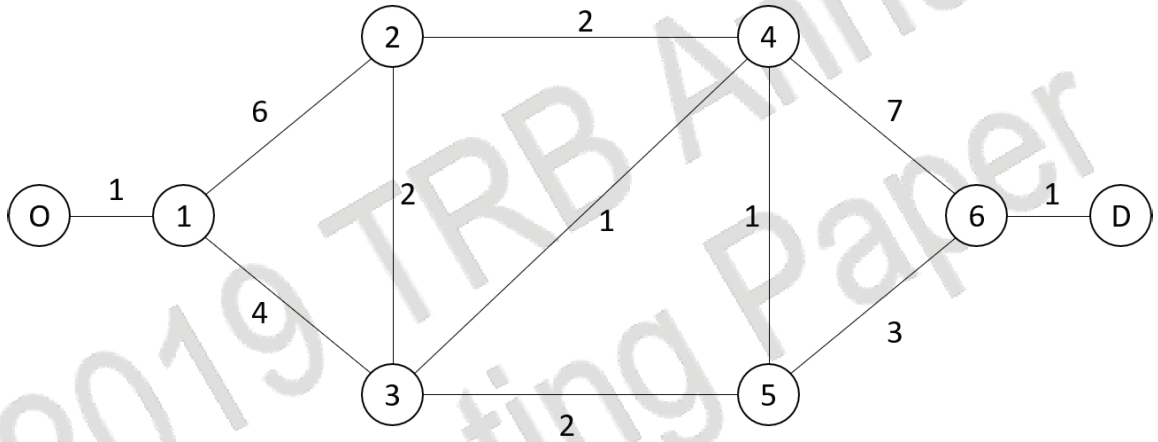

FIGURE 4 SIX-NODE NETWORK

3.1 Methods comparison

The comparison among methods is performed under 1000 different scenarios. In each scenario, the actual probabilities of *not* finding parking are randomly generated between [0.5, 1] for each parking facility, simulating a relatively congested parking condition. In each scenario, 100 parking search trajectories are generated from a purely random search policy as the training data used for learning. This is a reasonable data size considering the scale of a real special event.

We first compare the ranking (in terms of expected cost) of the strategies obtained by the three approximate methods: the greedy nearest-node strategy, the Q-learning method, and the proposed modified Q-learning method. From Table 1, it can be seen that the proposed modified Q-learning is ranked first (lowest expected cost) in 529 out of the 1000 scenarios; and is ranked last (highest expected cost) in only 99 scenarios. The ranking performances of the original Q-learning and the nearest-node strategy are similar, both inferior to the proposed modified Q-learning algorithm. Both of them are ranked first for only 200+ scenarios, but last for 400+ scenarios.

TABLE 1 COST RANKING COUNTS

Rank \ Strategy	Nearest	Q-learning	Revised Q-learning
First	228	243	529
Second	334	294	372
Third	438	463	99

We further benchmark the three approximate methods against the true optimal strategy solved using value iteration. Two performance metrics are examined: the average expected travel cost over the 1000 scenarios, and the performance profile (*32*).

In terms of the average expected travel cost over the 1000 scenarios, the true optimal strategy scores 11.56. The average expected cost of nearest strategy, Q-learning and modified Q-learning are 12.86, 12.92 and 11.77, respectively. The gap between the strategy obtained from the modified Q-learning and the optimal strategy is very small.

The performance profile provides a more detailed, graphical comparison of different algorithms over the same problem set (*32*). In this study, each scenario is a unique problem in the problem set. For each scenario c and algorithm m, $t_{c,m}$ is defined as the expected cost of the resulting strategy. We use the true optimal strategy as a baseline, which is always better or equal to other strategies but unknown to travelers. The performance ratio of algorithm m in scenario c is defined as:

$$r_{c,m} = \frac{t_{c,m}}{t_{c,optimal}}$$

We are interested in obtaining an overall assessment of the performance of the algorithms instead of any particular scenario. Now define

$$\rho_m(\tau) = \frac{n_c}{|C|}, c \in C: r_{c,m} \leq \tau$$

where n_c is the number of scenarios in which the algorithm m has a performance ratio within a factor τ of the best possible strategy. As defined, $\rho_m(\tau)$ is essentially the cumulative distribution function of the performance ratio of algorithm m. For example, a point $(1.05, 0.93)$ on the dotted line in Figure 5 indicates that the modified Q-learning algorithm can solve 93% problems in the problem set with a cost smaller than or equal to 1.05 times what the optimal solution takes.

It can be observed from Figure 5 that for the easier 40% of the 1000 scenarios, all three approximate algorithms can lead to strategies that have almost the same expected cost as the true optimal strategy (all three curves are almost vertical in the box defined by $\tau \in [1,1.1]$ and $\rho(\tau) \in [0, 0.4]$). On the other hand, for all of the 1000 scenarios, the proposed modified Q-learning would result in strategies that cost no larger than 2 times what the true optimal strategy costs (the $\rho(\tau)$ value reaches 1 before τ gets greater than 2 for the dotted line). But the other two approximate algorithms could lead to strategies that cost much more compared to the true optimal strategy. In the worst case, the estimated optimal strategies obtained from Q-learning could take up to 7.7 times the true optimal cost (the long tail of the solid curve hits $\tau = 7.7$ when $\rho(\tau) = 1$). This might be because the training data set in this case study is limited to 100 search trajectories. The

values of the original Q-functions are not properly trained with limited data. In the next section, we will explore the sensitivity of the Q-learning and the proposed limited Q-learning algorithms.

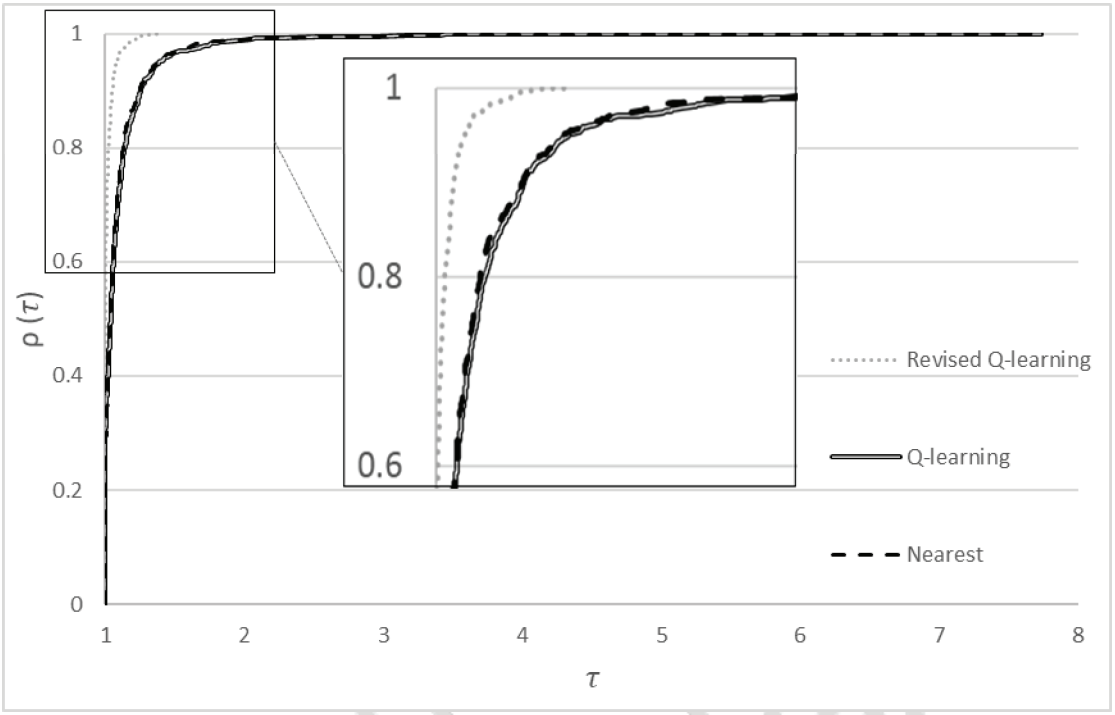

FIGURE 5 PERFORMANCE PROFILE OF CASE STUDY

3.2 Sensitivity analysis

Applying the same settings as in section 3.1, the purpose of this experiment is to investigate how the size of training data would affect the performance of Q-learning and revised Q-learning. The performance measure here is the relative excessive cost (REC), where the optimal cost from value iteration is used as benchmark.

TABLE 2 RELATIVE EXCESSIVE COST

DATA SIZE (Number of Random Search Trajectories)	**RELATIVE EXCESSIVE COST**	
	Q-Learning	Modified Q-learning
10	17.23%	12.46%
50	15.36%	3.20%
100	11.72%	1.83%
200	6.62%	1.75%
500	2.14%	1.12%
1000	0.71%	0.70%
10000	0.22%	0.28%

From Table 2, it can be seen that as the size of the training data set increases, the performance of the Q-learning gradually catches up and later surpassed modified Q-learning when more than 1000 search trajectories are involved. However, the parking conditions could have changed by

the time enough data is collected and an estimated user-optimal strategy is pushed to the drivers. The proposed modified Q-learning, on the other hand, is able to generate searching strategies close enough to the true optimal strategy with as few as 50 search trajectories.

4. Conclusion and Future Work

This paper investigates the idea of introducing learning algorithms into parking guidance and information systems that employ a central server, in order to provide estimated user-optimal parking searching strategies to individual travelers. The central server collects data from numerous travelers' parking search experiences in the same area within a reasonable time window, computes approximated user-optimal parking searching strategy, and distributes the resulting strategy to travelers who are still searching in the same time window. This cycle repeats for each time window. A modified Q-learning approach is proposed. The proposed learning algorithm takes advantage of the network topology and dramatically reduces the size of the problem. Our numerical experiments have demonstrated that the proposed algorithm is able to produce high quality solutions with limited training data, and thus has great potential to be applied in real time.

There are several interesting future research directions. The determination of reasonable time windows is not addressed in this paper. We briefly discussed possible approaches in Section 1. Further development and analysis of these approaches would contribute to a more comprehensive parking guidance and information service framework towards implementation. It would also be interesting to see how the different approaches to handling recommendation updates discussed in Section 1 would affect the performance of the system. Improvements to the learning algorithm itself might also be a possibility. In this paper, the centralized server is considered to have no *a priori* knowledge of the probabilities of finding parking. In some circumstance, we may be able to retrieve some *a priori* knowledge from available information even before training starts. This could further reduce the amount of data required and the proposed algorithm will converge faster to the user-optimal strategy. Another possibility is to relax the fixed probability within a time window into a time-dependent probability. No matter how fast a server could learn, the parking conditions may change constantly. How to generate time-dependent searching strategies from learning needs further investigation. Finally, the authors plan to incorporate other components into the parking network such as parking reservation, pricing and ridesharing and analyze their impact on the transportation network.

Acknowledgement

This research is supported by National Science Foundation projects CMMI 1363244 "Collaborative Research: Modeling and Analysis of Advanced Parking Management for Congestion Mitigation" and CMMI 1643175 "EAGER: A Living Lab for Smartphone-Based Parking Management Services". We would like to thank the anonymous reviewers for their insightful comments that helped improve this paper.

Author Contribution Statement

The authors confirm contribution to the paper as follows: study conception and design: J. Xiao, Y. Lou; data collection: J. Xiao; analysis and interpretation of results: J. Xiao, Y. Lou; draft manuscript preparation: J. Xiao, Y. Lou. All authors reviewed the results and approved the final version of the manuscript.

References

1. Shoup, D. C. Cruising for Parking. *Transport Policy*, Vol. 13, No. 6, 2006, pp. 479–486. https://doi.org/10.1016/j.tranpol.2006.05.005.

2. Van Ommeren, J. N., D. Wentink, and P. Rietveld. Empirical Evidence on Cruising for Parking. *Transportation Research Part A: Policy and Practice*, Vol. 46, No. 1, 2012, pp. 123–130. https://doi.org/10.1016/j.tra.2011.09.011.

3. Giuffrè, T., S. M. Siniscalchi, and G. Tesoriere. A Novel Architecture of Parking Management for Smart Cities. *Procedia - Social and Behavioral Sciences*, Vol. 53, 2012, pp. 16–28. https://doi.org/10.1016/j.sbspro.2012.09.856.

4. Leclercq, L., A. Sénécat, and G. Mariotte. Dynamic Macroscopic Simulation of On-Street Parking Search: A Trip-Based Approach. *Transportation Research Part B: Methodological*, Vol. 101, 2017, pp. 268–282. https://doi.org/10.1016/j.trb.2017.04.004.

5. Geroliminis, N. Cruising-for-Parking in Congested Cities with an MFD Representation. *Economics of Transportation*, Vol. 4, No. 3, 2015, pp. 156–165. https://doi.org/10.1016/j.ecotra.2015.04.001.

6. Cao, J., and M. Menendez. System Dynamics of Urban Traffic Based on Its Parking-Related-States. *Transportation Research Part B: Methodological*, Vol. 81, No. August, 2015, pp. 718–736. https://doi.org/10.1016/j.trb.2015.07.018.

7. Benenson, I., K. Martens, and S. Birfir. PARKAGENT: An Agent-Based Model of Parking in the City. *Computers, Environment and Urban Systems*, Vol. 32, No. 6, 2008, pp. 431–439. https://doi.org/10.1016/j.compenvurbsys.2008.09.011.

8. Levy, N., M. Render, and I. Benenson. Spatially Explicit Modeling of Parking Search as a Tool for Urban Parking Facilities and Policy Assessment. *Transport Policy*, Vol. 39, 2015, pp. 9–20. https://doi.org/10.1016/j.tranpol.2015.01.004.

9. Idris, M. Y. I., Y. Y. Leng, E. M. Tamil, N. M. Noor, and Z. Razak. Car Park System: A Review of Smart Parking System and Its Technology. *Information Technology Journal*, Vol. 8, 2009, pp. 101–113.

10. Rajabioun, T., and P. A. Ioannou. On-Street and Off-Street Parking Availability Prediction Using Multivariate Spatiotemporal Models. Vol. 16, No. 5, 2015, pp. 2913–2924.

11. Xiao, J., Y. Lou, and J. Frisby. How Likely Am I to Find Parking? – A Practical Model-Based Framework for Predicting Parking Availability. *Transportation Research Part B: Methodological*, Vol. 112, 2018, pp. 19–39. https://doi.org/10.1016/J.TRB.2018.04.001.

12. Geng, Y., and C. G. Cassandras. A New "Smart Parking" System Infrastructure and Implementation. *Procedia - Social and Behavioral Sciences*, Vol. 54, No. October 2012, 2012, pp. 1278–1287. https://doi.org/10.1016/j.sbspro.2012.09.842.

13. Rajabioun, T., B. Foster, and P. Ioannou. Intelligent Parking Assist. *2013 21st Mediterranean Conference on Control and Automation, MED 2013 - Conference Proceedings*, 2013, pp. 1156–1161. https://doi.org/10.1109/MED.2013.6608866.

14. Ibeas, A., L. dell'Olio, M. Bordagaray, and J. de D. Ortúzar. Modelling Parking Choices Considering User Heterogeneity. *Transportation Research Part A: Policy and Practice*, Vol. 70, 2014, pp. 41–49. https://doi.org/10.1016/j.tra.2014.10.001.

15. Chaniotakis, E., and A. J. Pel. Drivers ' Parking Location Choice under Uncertain Parking Availability and Search Times : A Stated Preference Experiment. *Transportation Research Part A*, Vol. 82, 2015, pp. 228–239. https://doi.org/10.1016/j.tra.2015.10.004.

16. Asakura, Y., and M. Kashiwadani. Effects of Parking Availability Information on System Performance: A Simulation Model Approach. *Proceedings of VNIS'94 - 1994 Vehicle Navigation and Information Systems Conference*, 1994, pp. 251–254. https://doi.org/10.1109/VNIS.1994.396832.

17. Tang, S., T. Rambha, R. Hatridge, S. D. Boyles, and A. Unnikrishnan. Modeling Parking Search on a Network Using Stochastic Paths with History Dependence. *Transportation Research Record*, Vol. 2467, No. 1, 2014, pp. 73–79.

18. Abdulhai, B., R. Pringle, and G. J. Karakoulas. Reinforcement Learning for True Adaptive Traffic Signal Control. *Journal of Transportation Engineering*, Vol. 129, No. 3, 2003, pp. 278–285. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:3(278).

19. Arel, I., C. Liu, T. Urbanik, and A. G. Kohls. Reinforcement Learning-Based Multi-Agent System for Network Traffic Signal Control. *IET Intelligent Transport Systems*, Vol. 4, No. 2, 2010, p. 128. https://doi.org/10.1049/iet-its.2009.0070.

20. Prashanth L. A., and Shalabh Bhatnagar. Reinforcement Learning With Function Approximation for Traffic Signal Control. *Intelligent Transportation Systems, IEEE Transactions on*, Vol. 12, No. 2, 2011, pp. 412–421. https://doi.org/10.1109/TITS.2010.2091408.

21. Cai, C., C. K. Wong, and B. G. Heydecker. Adaptive Traffic Signal Control Using Approximate Dynamic Programming. *Transportation Research Part C: Emerging Technologies*, Vol. 17, No. 5, 2009, pp. 456–474. https://doi.org/10.1016/j.trc.2009.04.005.

22. Šemrov, D., R. Marsetič, M. Žura, L. Todorovski, and A. Srdic. Reinforcement Learning Approach for Train Rescheduling on a Single-Track Railway. *Transportation Research Part B: Methodological*, Vol. 86, 2016, pp. 250–267. https://doi.org/10.1016/j.trb.2016.01.004.

23. Arentze, T. A., and H. J. P. Timmermans. Modeling Learning and Adaptation Processes in Activity- Travel Choice: A Framework and Numerical Experiments. *Transportation*, Vol. 30, No. 1, 2003, pp. 37–62. https://doi.org/http://dx.doi.org/10.1023/A:1021290725727.

24. Desjardins, C., and B. Chaib-draa. Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, No. 4, 2011, pp. 1248–1260. https://doi.org/Doi 10.1109/Tits.2011.2157145.

25. Dai, X., C. K. Li, S. Member, A. B. Rad, S. Member, A. B. Rad, and S. Member. An Approach to Tune Fuzzy Controllers Based on Reinforcement Learning for Autonomous

Vehicle Control. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 6, No. 3, 2005, pp. 285–293. https://doi.org/10.1109/TITS.2005.853698.

26. Rummery, G. A., and M. Niranjan. On-Line Q-Learning Using Connectionist Systems. *Cambridge, England: University of Cambridge, Department of Engineering*, Vol. 37, 1994.

27. Sutton, R. S., and A. G. Barto. *Reinforcement Learning: An Introduction.* Cambridge: MIT press., 1998.

28. Watkins, C. J. C. H. *Learning from Delayed Rewards*. 1989.

29. Schwartz, A. A Reinforcement Learning Method for Maximizing Undiscounted Rewards. *Machine Learning Proceedings 1993*, Vol. 298, 1993, pp. 298–305. https://doi.org/10.1016/B978-1-55860-307-3.50045-9.

30. Federgruen, A., P. J. Schweitzer, and H. C. Tijms. Contraction Mappings Underlying Undiscounted Markov Decision Problems. *Journal of Mathematical Analysis and Applications*, Vol. 65, 1978, pp. 711–730.

31. Melo, F. S. Convergence of Q-Learning: A Simple Proof. *Institute Of Systems and Robotics, Tech. Rep*, 2001, pp. 1–4. https://doi.org/10.1016/j.aqpro.2013.07.003.

32. Dolan, E. D., and J. J. Moré. Benchmarking Optimization Software with Performance Profiles. *Mathematical Programming, Series B*, Vol. 91, No. 2, 2002, pp. 201–213. https://doi.org/10.1007/s101070100263.