Journal of Network and Computer Applications 87 (2017) 210–222

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

MDP: Minimum delay hot-spot parking

Peng Liu[a], Biao Xu[a], Guojun Dai[a], Zhen Jiang[b,c,*], Jie Wu[b]

[a] Institute of Computer Application Technology, Hangzhou Dianzi University, China
[b] Department of Computer and Information Sciences, Temple University, United States
[c] Department of Computer Science, West Chester University, United States

A R T I C L E I N F O

Keywords:
Hungarian algorithm
Parking guidance information system (PGI)
Traffic performance optimization
Vehicular ad-hoc network (VANET)
Wireless communication

A B S T R A C T

Hot-spot parking is becoming the Achilles' heel of the tourism industry. The more tourists that are attracted to the scenic site, the more often they will encounter a hassle of congestion to find a parking place; while those existing facilities for daily traffic are not supposed to support the excessive volume outburst. In this paper, we present a new parking guidance information system (PGI). By taking advantage of the technical advances of today in wireless communication of vehicular ad-hoc network, each vehicle will request and obtain a relatively fair opportunity to park. The competition and the corresponding allocation on the available slots emerging along the time scale are considered, in order to ensure that no vehicle enters a state of starvation. This is the first attempt to solve the spatiotemporal problem of resource assignment based on our extensive work on the Hungarian algorithm. The contribution as one part of the sustainable development of big historic cities is to minimize the idle driving and waiting, without increasing the parking supply, which could be costly and unnecessary to build in those urban areas. Both analytical and experimental results demonstrate the success of our effort, in terms of the average cruising/waiting time in each individual parking case and its upper bound. The data is compared with the best results known to date and shows a new direction to improve the resource assignment.

1. Introduction

West Lake was made the UNESCO World Heritage Site in 2011 (Espanol, 2011). It has been the best-known hot-spot over centuries to attract many tourists. But during the travel season, such as the Golden Week Holiday, a high parking volume usually exceeds the capability of existing facilities, incurring the so-called hot-spot parking problem (e.g., theexpiredmeter, 2010).

The delay in searching and occupying a parking slot might cause congestion and environmental issues as indicated in Geng and Cassandras (2013). The time includes the period a vehicle drives towards the target place according to the reservation. It also includes the idle driving around the scenic site when the vehicle waits for the vacant slot to emerge. In our hot-spot parking, such a delay has a direct impact on municipal reputation and revenue, while tourism has become one of the world's fastest growing industries as well as the major source of earning and employment for many developing countries.

Unlike the problem of residential parking (No parking, 2012) that can resort to new construction of parking facilities (Millikin, 2013), this is a resource allocation problem, but in an extremely critical circumstance where those slots constituted for daily traffic are required to allocate for the volume outburst (e.g., Yang, 2014). When the slots currently available are not enough to support all parking demands, the capacity of each place growing along the time scale must be considered for the vehicle to capture the future parking opportunity. This introduces the spatiotemporal resource allocation problem discussed here.

Many existing parking guidance information systems (PGI), either reservation based (e.g., Geng and Cassandras, 2013) or greedy (e.g., Ayala et al., 2012), provide parking guidance by allowing every vehicle to reach the nearest available slot. However they overlook the competition of limited slots in the resource-critical scenarios and cannot provide a fair chance for those runner-ups to reenter the parking competition. The corresponding slot allocation to the closest vehicle ignores the fact that many vehicles behind it are runner-ups from the early parking competitions, which have cruised for a long time since they lost the parking opportunities. Due to the distance existing for a runner-up to approach the next available slot, its priority of parking reservation will be depleted by any vehicle ahead that newly joins the competition. Such a depletion will force those runner-ups back to the idle driving. As often seen in the reality, when an overwhelming amount of vehicles frequently join the parking competition from everywhere, one loss usually leads to a sequence of

* Corresponding author at: Department of Computer Science, West Chester University, United States.

http://dx.doi.org/10.1016/j.jnca.2017.03.002
Received 4 October 2016; Received in revised form 2 February 2017; Accepted 5 March 2017
Available online 08 March 2017

consecutive losses and even a starvation.

Remark. Considering that the above starvation creates an endless delay effect, the aforementioned spatiotemporal resource assignment (dispatching all m vehicles to n places where $m >> n$) becomes non-trivial. Existing PGI strategies (e.g., Ayala et al., 2012; Benenson et al., 2008; Geng and Cassandras, 2013; Jin et al., 2012) or similar assignment-based schemes for vehicle dispatching (e.g., Alfonsetti et al., 2015; Gao et al., 2016; Maciejewski et al., 2016; Miao et al., 2016) are applied on a sufficient number of targets only. As we will demonstrate later, the starvation cannot be avoided completely when the demand exceeds the supply.

In this paper, we present a solution for the PGI system under its common structure (e.g., Geng and Cassandras, 2013; Wan et al., 2014). The proposed assignment of parking slots is derived from the Hungarian Algorithm (Bondy and Murty, 1976). We first consider the extension from the solution for the traditional quadratic assignment problem (QAP) (Wikipedia,). Such a weighted bipartite matching takes both weights on edges and vertices but does not increase any time complexity as we will prove it later. Then we present our assignment by utilizing the capacity growth along the time scale. The corresponding complexity is bounded within a linear-time incremental structure, in order to achieve a practical system implementation. As the result, the total time needed for all parking processes (i.e., the average of an individual case) can be minimized in both the above assignment solutions. Meanwhile, the worst case can be bounded within a certain period. Such minimum parking scheme is denoted by MDP. Its key is to capture the potential competition along each vehicle's trajectory, the corresponding cruising cost to the next available slot, and then those future competitions along each possible path in a heuristic manner.

2. Methodology and contribution

We studied the unique feature of the hot-spot parking in big historic cities where any new construction of parking facilities is usually expensive (e.g., Millikin, 2013) or unnecessary for the off-peak traffic (e.g., Blog,). The focus was on the scenarios under critical resource constraint where the endless delay effect of the starvation problem cannot be avoided completely in the existing PGI systems or with similar vehicle dispatching schemes (as explained later in Section 3). Our goal was to control the cruising/waiting of each parking process within a certain bound, while the total time for the scheduled vehicles to stay in traffic can be optimized.

This leaded to the optimization in a spatiotemporal assignment MDP (as described in Section 5). The well-known Hungarian Algorithm for the QAP with the maximum-weight assignment is the preliminary (as shown in Section 4). Our solution was derived from the extension of the Hungarian Algorithm in additional scales, under a linear-time incremental structure: the first is for allocation with enough vacant slots and second is based on the prediction of the growth of parking capacity in the time scale, which can be better suitable for those extreme cases of slot competitions in our hot-spot parking. The technical details can be seen in Section 6.

We developed the simulation with the real traffic data from the road tests. The experimental results (as presented in Section 7) verified the substantial improvement of our approach MDP in terms of the elapsed time on both the worst case and the average case, compared with the results from the existing PGI services (Ayala et al., 2012; Geng and Cassandras, 2013). Thus, the extra cruising time and volume of traffic caused by the delay in finding a place to park can be minimized. The corresponding congestion and environment issues can be mitigated as one part of sustainable development in the city Hangzhou. At the end of this paper, Section 8 provides the conclusion and ideas for future research.

3. Related work and our research incentives

First, our work targets a resource allocation problem. Compared with existing reservation systems for parking, our MDP faces the challenge of extreme lack of resources when the existing facilities limited for the off-peak volume are used to support the high volume during the peak time. It is costly (e.g. Millikin, 2013) and unnecessary (e.g., Blog) to directly build new parking facilities at the level to accommodate the volume outburst. Such a development can possibly induce more traffic and worsen the problem (Blog,). The recent improvement achieved on assisting the parking, such as occupancy increment (e.g., Abdullah et al., 2012; Baroffio et al., 2015; Salpietro et al., 2015) and ease of parking operation (e.g., Young, 1991; Yan et al., 2011), cannot reduce any conflict and further mitigate the delay impact. The hot-spot usually attracts many people so that any effort based on congestion control (e.g., Ayala et al., 2012; Pierce and Shop, 2013) will go in vain.

Secondly, the assignment of parking slots is expected to reduce the total cruising/waiting time of vehicles, in order to minimize their unnecessary dwellings in the traffic. This will mitigate the congestion and environmental issues that are associated with our hot-spot parking delay problem. As we expect, when a slot becomes available in front of any two vehicles, the one with less elapsed time in its successive cruising after the back-off will sacrifice and defer the parking request. Therefore, our reservation requires the complete information of all vehicles' trajectories, the corresponding cruising cost to their next vacant slots, and then those potential competitions along each of the cruising paths.

Thirdly, we need the bound of each vehicle's cruising or waiting. Without our comprehensive view of the slot competition loss and the corresponding cruising cost, a vehicle may be sacrificed and pushed away back to the idle driving, not being able to seize the target slot as it is supposed to be. As a consequence of high demand against insufficient parking supply, it may take a long time for the next vacant slot to appear along the trajectory of such a vehicle. In the usual case, this newly-emerged available slot will soon be depleted again by other vehicles nearby when this delayed vehicle is still far away in its cruising. In the worst situation, that vehicle can encounter a starvation and falls into an endless loop of missed, cruising, missed again.

Unique circumstance in the cruising of the hot-spot parking. We observed that cyclic route is commonly used in the cruising. Many users of our MDP system are tourists. Their search for parking places is limited due to the lack of sufficient local information. Unlike a spiral-like search for the parking places, which forces to gradually leave from the travel destination, the cyclic route helps to seize the parking opportunity around the scenic site. Fig. 1 (a) shows the sample routine that is recommended by the travel agent and GPS for the tourists to go to the most popular sites, Sudi and Feilai Peak around the West Lake in the one-day trip during the Golden Week. In the real traffic, this routine and the driving directions are predetermined by the local government to mitigate the volume traffic or jams (in Fig. 1 (b)). This traffic model also helps to simplify our discussion in the paper. It is noted that our solution is also applicable for bidirectional traffic because the vehicle trajectory is represented by the arrival time at each parking place in our algorithms, supporting vehicle driving in every direction or even changing the direction at any time. The sample is applied for vehicles intending to park at those top 5 facilities only. Those five places are denoted by a, b, c, d, and e, respectively. Note that we focus on a technical solution here. An accreditation system will be associated with the real system implementation in order for every user to follow the guidance, and not to create the interfering noise by changing the routine or falsifying the data in the parking request. However, that part is omitted here due to the scope of this paper.

Unlike residential parking, in which one may need the permission for overnight parking, a 3-h limit is commonly adopted in many urban areas for the hot-spot parking. Therefore, the capability growth along

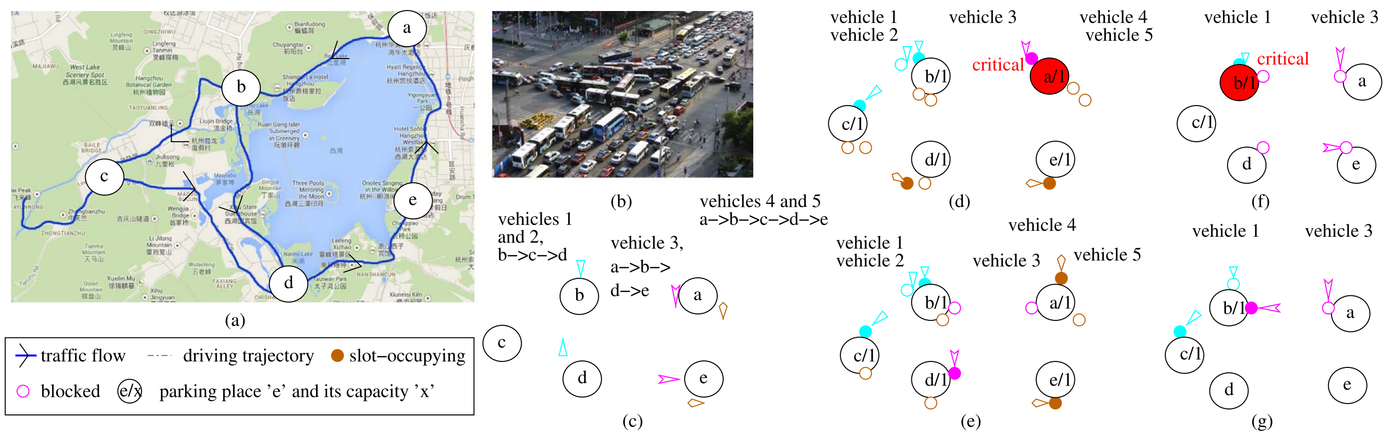


Fig. 1. (a) Map of the West Lake scenic area with the top 5 parking facilities (List of parking, 2006). (b) Volume traffic in holiday season caused by insufficient parking management of private vehicles. (c) 5 vehicles in parking requests and their trajectories. (d) Cruising and slot occupancy with the existing shortest-path-based reservation where the critical slot reservation has been highlighted in red. (e) Target optimization on total cruising time after a switch of slot assignment at site-a (explained later in Section 6.1). (f) A possible starvation with an incorrect slot resrvation/assignment at the site b in red. (g) MDP solution for the starvation problem (explained later in Section 6.2).

the time scale can be predicted with recent surveillance technologies (e.g., Zheng et al., 2015), helping us to control the cruising within a desired bound.

Starvation as the parking delay problem. In the following, we use a real scenario to explain the incentive of our research. In the first example, we demonstrate our target optimization on the total cruising time and its need for the information of all competitions along the vehicle trajectories. In the second example, we illustrate the use of such heuristic path information to solve the starvation problem. First, we show the shortage of using the shortest path as the metric in choosing the slot assignment in existing PGI schemes (e.g., Ayala et al., 2012; Benenson et al., 2008; Geng and Cassandras, 2013). Then, we show the limit in similar vehicle dispatching (e.g., Alfonsetti et al., 2015; Gao et al., 2016; Maciejewski et al., 2016; Miao et al., 2016) that adopt the assignment on slots currently available only. The starvation problem cannot completely solved and its endless delay effect cannot be overlooked; even the centralized resource such as cloud (e.g., Arif et al., 2012; Geng and Cassandras, 2013; Salpietro et al., 2015; Wan et al., 2014) is adopted. This induces the need for a new and complete solution.

In the first example, each parking place has one vacant slot, and five vehicles are on schedule. Two local drivers, denoted by 1 and 2, adopt route $b \to c \to d$ to enter the area (see the cyan path in Fig. 1 (c)). Vehicle 3 comes from the suburban area and the driver is familiar with the traffic situation. Only the tour around the lake $a \to b \to d \to e$ (see the magenta path in Fig. 1 (c)) is needed. The other two drivers, denoted by 4 and 5, come from a nearby city and do not have parking site preferences. The route $a \to b \to c \to d \to e$ (see the brown path in Fig. 1 (c)) is predetermined in order to obtain any place around the lake.

In the traditional PGI (e.g., Ayala et al., 2012; Benenson et al., 2008; Geng and Cassandras, 2013; Jin et al., 2012) based on the shortest path, vehicle 3 will take the slot at site a, while vehicles 1 and 2 will take the slots at sites b and c, respectively. Such occupancy (see Fig. 1 (d)) will force vehicle 4 to go to site d. If we switch the allocation between vehicles 3 and 4, vehicle 3 can take the shortcut $b \to d$ (see the critical site in Fig. 1 (e)), saving the time of vehicle 4 along $b \to c \to d$. The target optimization on the total cruising time (explained later in Section 6.1 in our MDP solution) requires to consider the entire path of vehicle 4 when it competes against vehicle 3 at site a (see the red site in Fig. 1 (d)).

In the second example, we show that such a problem can go worse where the victims are entrapped into a starvation situation, even when a complete slot allocation is still feasible in the global view. With the same setting of routes as the above example, we consider vehicles 1 and 3 only in Fig. 1 (f)). We assume that only sites b and c each has one vacant slot. In the aforementioned shortest-path-based scheme, vehicle 1 can be selected for site b, leaving vehicle 3 in the starvation (see all unfilled circles as the block signs along the trajectory in Fig. 1 (f)). Had we known at the global view level that site b is the only option for vehicle 3 along the entire cruising path, vehicle 1 can back-off at site b (see the red site in Fig. 1 (f)). Thus, the parking problem can be solved (see the explanation in Section 6.2 for Fig. 1 (g)). But this requires the heuristic path information of each vehicle.

In an $m \times n$ assignment-based scheme (e.g., Alfonsetti et al., 2015), m vehicles can be dispatched to n customers when $n \geq m$. But when $m > n$, the same scheme repeated for the rest $(m - n)$ vehicles cannot avoid the starvation problem. For instance, we consider vehicles 1 and 3 in Fig. 1 (g). But this time, only site b has one vacant slot. The vacancy at site c will occur later by the time vehicle 1 approaches it. That is, the above optimization in Fig. 1 (g) is still feasible (see the discussion on our MDP solution in Section 6.2). Considering two vehicles' competition at site b, any existing 2×1 assignment (e.g., Gao et al., 2016; Maciejewski et al., 2016) will dispatch vehicle 1 first. Then, by the time the vacancy at site c occurs, it is now unreachable for vehicle 3. That is a starvation because the location and capability growth of site c is unknown to the previous assignment at the beginning. The distance weight to each of m vehicles is not determined. This induces the need for our spatiotemporal assignment, which will be discussed in the rest of this paper.

4. Preliminary

The quadratic assignment problem (QAP) (Wikipedia,) is one of the fundamental problems in the branch of optimization. It models the following real-life problem: There are n agents ($\in X$) and n tasks ($\in Y$). Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task and exactly each task to each agent so that the total cost of the assignment is minimized.

Algorithm 1. Hungarian algorithm (Bondy and Murty, 1976) for assigning n agents ($\in X$) to n tasks ($\in Y$) in a QAP.

1) **Initialization**. For each agent i ($\in X$) and a possible task j ($\in Y$), initiate $R(i, j) = -c_{ost}(i, j)$ where $c_{ost}(i, j)$ is the corresponding cost (for agent i to accomplish task j). After that, set $L(v)$ with Eq. (1).
2) **Completion check**. If every agent has the reservation, stop the algorithm; otherwise, for any unassigned agent x, initiate

$S = \{x\}$, $T = \{\}$, and the E-tree $E_x^* = \{x\}$ (also called the alternating tree in Bondy and Murty (1976)). Then, go to the next phase.

3) **Label update for any possible assignment**.
 • If $N(S) \neq T$, go to phase 4. $N(S)$ is the set of tasks that have been assigned or to be able to match with any agent in S; that is, $\{j|(i, j) \in E_x^* \wedge i \in S\}$.
 • Otherwise, calculate α with Eq. (2) and then update L with Eq. (3). After that, keep the edges of those assignments in tree E_x^* and add the new edge (i,j) when $L(i) + L(j) = R(i, j)$ (according to the new L labels). Then, go to the next phase.

4) **Construction from any** $y \in N(S) - T$.
 • If y has not been assigned, alternate the assignments from i (found in the above phase 3) to the root x, along the so-called augmenting path in E_x^* (denoted as the E-path ${E_x}^i$ in this paper). After that, add the edge (i,y) in E_x^* as the new assignment and go to phase 2.
 • Otherwise, y has been assigned with an agent, say z in E_x^*. Add the edge (z, y) to E_x^* as a record, and set $S = S \cup \{z\}$, $T = T \cup \{y\}$. After that, go to phase 3.

Such an assignment problem can be solved with the Hungarian algorithm (Bondy and Murty, 1976) in four phases. The details are shown in Algorithm 1. Basically, each bipartite matching between agents and tasks is conducted from the lowest cost (see the first part of phase 4). The conflict is solved within a time bound. Such a bound is denoted by $\triangle$ and can be calculated as $\max\{-L(v)|v \in S\}$, where L is the labeling function in Bondy and Murty (1976) and S denotes the considered agents in the record. For any possible assignment (i, j) under the consideration S, those relevant assignments in the existing bipartite matching will be shuffled, in order to add the task j and complete the current bipartite matching that was initiated from agent x. To find such an augmenting path in an easy way, the implementation in x ray () is adopted and we can deal with it as a maximum-weight matching problem by using a non-positive value to express each cost (see phase 1 of Algorithm 1). Thus, the total cost in the assignment within the consideration under the time bound $\triangle$ can be optimized. As this bound $\triangle$ increases greedily (in phase 3), more agents will join the matching (see the end at phase 2). The above process will be repeated, until all agents can be assigned. At the end, the desired optimization (on total cost in assignment) can be obtained.

$$L(v) = \begin{cases} \max_{y \in Y} R(v, y), & v \in X \\ 0, & v \in Y \end{cases} \tag{1}$$

$$\alpha = \min_{x \in S, y \in Y-T}\{L(x) + L(y) - R(x, y)\} \tag{2}$$

$$L(v) = \begin{cases} L'(v) - \alpha, & \text{if} v \in S \\ L'(v) + \alpha, & \text{if} v \in T \\ \text{no change}, & \text{otherwise} \end{cases} \tag{3}$$

In Algorithm 1, the agents and the tasks that have been considered previously can be found in the records of S and T respectively. $N(S) \neq T$ implies the existence of a new assignment that has not been considered. If such a task y has been assigned at a lower price to another agent z, we can merge the result (z, y) with those under the current search in E_x^* (in the second part of phase 4). Otherwise, we have $i \in X$ and $y \in N(\{i\}) \wedge y \notin T$. (i, y) will be assigned immediately (in the first part of phase 4) because i has the lowest price of y among all agents. The previous assignment on i, if any, will be shuffled. This switch process will continue in E_x^* until it stabilizes at the root x, along the path from x to i that is denoted as the E-path ${E_x}^i$. As a result, all agents in S will be in a perfect bipartite matching. The above merging and shuffle processes in phase 4 will be repeated until $N(S) = T$. After that, the current bound $\triangle$ will be recalculated in phase 3 in order to consider a higher price to solve the conflict in the matching. The entire procedure can converge when every agent has its own assignment. The detailed sample of this shuffle and merging can be seen in x ray (), and they are also demonstrated in the later discussion on our Hungarian-algorithm-based MDP solution.

Theorem 1 (*x ray () the optimization achieved by Alg. 1 and its time complexity*). *Algorithm 1 will end with a bipartite matching so that the total cruising time can be minimized. Its overall complexity $T^1(n)$ is $O(n^4)$, where $n = |Y|$.*

Proof. This well-known claim of the Hungarian and its proven can be found in a lot of existing work (e.g., Bondy and Murty, 1976 and x ray,). □.

The above QAP includes every agent and task as the vertices in the bipartite matching, but each appears only once. When we consider the vehicle as agent and the parking place as the task in the slot allocation problem, a parking place needs to match with multiple vehicles as its capacity allows. The capacity of a parking place will be taken as the weight of the vertices. The corresponding merging and shuffle processes will be considered as a special weighted bipartite matching (e.g., Gao et al., 2016), where the weights on both edges and vertices are taken. From the next section, based on Algorithm 1, we will present our MDP solution and its extension along the time scale.

5. System setting and problem formation

Our proposed system, MDP, takes the basic structure of PGI as Geng and Cassandras (2013). The allocation center will provide a schedule to serve all parking places Y in the entire area around the travel hot-spot. The driver will select the attraction site as the travel destination and those nearby parking places. After that, such information will be sent to the center via the wireless communication, with the parking reservation request. Note that each request is sent independently. Then, the system will respond with a reserved parking place. By approaching this target (parking place) along its predetermined trajectory, each vehicle, denoted by $x \in X$, can ensure the speed (Asif et al., 2014) and driving time (Ganti et al., 2014) within an acceptable range. Meanwhile, the total elapsed time of all vehicles in scheduling can be optimized to the minimum, in each vehicle's driving in the distributed manner. Table 1 summarizes all of the notations used in this paper, which will be explained in the following.

At each parking place $y \in Y$, the vacancy can be detected (e.g., Choeychuen, 2013) and is denoted by C_y. The access of parking is verified in a vehicle-to-roadside (V2R) communication. It can be granted at the gate only to the driver who has received the e-ticket in our reservation response, but the gate is not necessary to know which specified slot. Thus, the capacity increases as vehicles leave. For the scheduling conducted at t=0, such a change at time $t > 0$, denoted as ${C^t}_y$, can be ensured with our parking policy, or can be predicted with those history records on vehicle behaviors in a conservative manner (Zheng et al., 2015). Thus, we have $C_y^0 = C_y$ and $C_y^t \leq C_y^{t+\delta}$ for any period $[t, t + \delta]$.

Researchers have developed methods to predict driving speed (Asif et al., 2014) and travel time (Ganti et al., 2014), in order to solve the traffic issues during the vehicles cruising along their trajectories. By adopting the GPS or other global maps, the trajectory of each vehicle x, can be interpreted by its arrival time at every place $y \in Y$ (Sana et al., 2014), which is denoted by $R(x, y)$. As we addressed earlier, we deal with a maximum-weight matching problem and will have a non-positive value in each $R(x, y)$. Our system will select an arrival of vehicle x in $R(x, y)$ when y has at least one vacant slot ($C_y > 0$). Such an available selection is calculated and stored in table m to avoid overbooking. Respectively, the trajectory taken after time t is denoted by $R^t(x, y)$ and the selection on R^t is mapped into m^t, under the capacity constraint $C_y^t > 0$.

Our goal is to assign each vehicle $\in X$ to a parking place $\in Y$ with a

Table 1
Notations.

X	vehicle set $X = \{1, 2, 3, \cdots\}$
$\|X\|$	total number of vehicles ($\in X$) in schedule
Y	set of parking places $Y = \{a, b, c, \cdots\}$
$\|Y\|$	number of parking places
C_y	available space (also called parking capacity) of $y \in Y$
C_y^t	capacity of $y \in Y$ at time t, where any period $[t1, t2]$ has non-descending records $C_y^{t1} \le C_y^{t2}$
$R(x, y)$	cost Sana et al. (2014) for $x \in X$ to reach $y \in Y$ in terms of elapsed time where "-" indicates an initial/unreachable status
r_x	the cycle of vehicle x along its trajectory
$R^t(x, y)$	the $R(x, y)$ describing the cost for $x \in X$ to reach $y \in Y$ after time t along its cyclic route
$m(x, y)$	bipartite matching between $x \in X$ and $y \in Y$ where 1 denotes a saturated assignment, 0 denotes a possible assignment, and "-" is the initial status
$m^t(x, y)$	the above bipartite matching between $x \in X$ and $y \in Y$ for each capacity change C_y^t
$L(v)$	labeling function of Hungarian algorithm Bondy and Murty (1976), $v \in X \cup Y$
$L'(v)$	previous record of L for any given $v \in X \cup Y$
α	the difference between $L(v)$ and $L'(v)$ each time
$\vartriangle$	the upper bound of interval where the assignments have been considered, changeable as the process progresses, i.e., $\max\{-L(x) \| x \in X\}$
∇	the upper bound of the entire interval in consideration, changeable, in which each vehicle can cruise along its cyclic route to reach every possible place after time $\vartriangle$
S	vehicle set in the current consideration of allocation, $\subseteq X$
$N(S)$	places ($\subseteq Y$) that are assigned to or arrived by vehicles $\in S$, i.e., $\{j \| \exists\, m(i, j) = 0 \text{for arrival, or} 1 \text{for assigned}\}$, or $\{j^t \| \exists\, m^t(i, j) = 0 \text{or} 1\}$
T	set of assignments reserved, i.e., $\{j \| \exists\, m(i, j) = 1\}$ or $\{j^t \| \exists\, m^t(i, j) = 1\}$
@	status of a place ($y \in Y$), saturated (=0) or unsaturated (>0) for another vehicle under its capacity, i.e., $^{@}y = C_y - \sum_{x\in X} m(x, y)$ or $C_y^{\vartriangle} - \sum_{x\in X, 0\le t\le \vartriangle} m^t(x, y)$
E_u^*	an alternating tree Bondy and Murty (1976) derived from m, with the root u, simply called E-tree
E_u^v	an augmenting path Bondy and Murty (1976) in E_u^*, with u and v as end points, simply called E-path

limited cruising or waiting time, while the total cruising time (on the way to each target along their predetermined trajectories) can be minimized. The assignment at t=0 for the cruising trajectories afterward can be formalized as a maximum-weighted matching. With our extensive consideration on the capacity at each location and the growth of such capacity along the time scale, the problem is formalized in the following:

$$\underset{m^t}{\text{argmax}} \sum_{x\in X} \sum_{y\in Y} \sum_{0\le t<\infty} R^t(x, y) m^t(x, y)$$

$$s.\ t.\ \ \text{every } m^t(x, y) = 0, 1, \text{or "}-\text{"} \qquad i)$$

$$\sum_{y\in Y} \sum_{0\le t<\infty} m^t(x, y) = 1 \text{ for every } x \in X \qquad ii)$$

$$\sum_{x\in X} \sum_{0\le j\le t} m^j(x, y) \le C_y^t \text{ for every } y \in Y \qquad iii)$$

$$C_y^t \le C_y^{t+\delta} \text{ for any } \delta > 0 \qquad iv)$$

$$-R^t(x, y) \ge t \ge 0 \text{ upon the occurrence of } C_y^t \qquad v)$$

$$R^{t+\delta}(x, y) \le R^t(x, y) \text{ for every} \delta > 0 \qquad vi)$$

"-" indicates an initial or unreachable status and is calculated as "0". Constraint i) ensures the slot assignment as a bipartite matching. Constraint ii) guarantees such an assignment without double-booking. Constraint iii) asserts the use of slots under the capacity constraint, while constraint iv) indicates our assumption on the capacity increment. When C_y^t is replaced by a fixed value C_y, the above will represent our solution on the snapshot where there is enough vacancy for all vehicles on schedule. Constraint v) confirms the driving cost in the road and ensures the availability of the vacant slot until the vehicle with the reservation arrives. Constraint vi) secures the same time scale t for considering both the capacity growth C^t and the elapsed time of cruising R^t. Both constraints iv) and v) can be relaxed when our approach is applied to an open system with real time slot surveillance and prediction (e.g., Zheng et al., 2015).

Note that this, as we addressed in early Section 3, is a totally new problem with our consideration of R and C. The consideration of their changes along with the time t passing is also important and distinguishes our contribution from any existing methods. Such a time t cannot be limited in the interval $[0, \lambda]$ when the vacant slots are enough for all vehicles under the schedule, i.e., $\lambda = \min\{t \| \sum_{y\in Y} C_y^t \ge \|X\|\}$. When the last vacant slot appears too far away from the vehicle to reach, some capacity change soon emerging in neighborhood could be a better choice. The above proves the research incentive of our 2-stage development in this paper. The first is for the case when C is enough and the second is for the case when C can grow.

When demand exceeds supply, some vehicles cannot seize the reservation for obtaining an available slot. In order for them to obtain the second chance and to park close to their destination, in our approach, each of them will continue its cruising along a cyclic route until a reserved place can be reached. The justification of this cyclic cruising can be found in early Section 3 as one of our observations on the hot-spot parking. Thus, we have:

$$\begin{cases} R^{t+\delta}(x, y)(1 - \sum_{0\le j\le t} m^j(x, y)) \le r_x + R^t(x, y) \\ \text{for every } R^{t+\delta}(x, y) < R^t(x, y) \text{ or any} \delta > r_x \qquad vii) \end{cases}$$

From constraints vii), if the arrival $R^t(x, y)$ (of vehicle x at place y since time t) is not considered in assignment m^t (i.e., =0), the route in a cycle r_x can guarantee the second entry at the same place to appear within a bounded time after time-$(t + \delta)$. Note that this constraint can be relaxed when the cyclic route becomes not necessary. However, the impact is limited on the upper bounds of cruising time and information collection (see Theorem 5 in later discussion). It is because of the need for considering any possible place along the trajectory as long as the vehicle drives.

Our simulation work also studies the scenarios in which the vehicles are allowed to change the routine and to slowly cruise around the reserved place. In such an extension (denoted by MDP+), the delay in waiting for a vacant slot to occur, will be compared with the time spent in driving to the next place along the preselected trajectory. The schedule will be made to reduce the total end-to-end delay, while each individual parking process can still have a bounded performance.

6. The proposed solution MDP

This section provides our 2-stage development on the above maximum weight target, as our MDP solutions to reduce the total cruising/waiting in the hot-spot parking. The first is for the slot allocation at time t=0 with sufficient parking vacancy C, and then the second is for the allocation along the time scale. For each solution, we also provide the upper bound for each vehicle in its cruising or waiting.

The issues in the service implementation are also discussed.

6.1. Solution with enough vacant slots

We first focus on the MDP solution on a snapshot with sufficient slot vacancy in the global view. The problem can be simplified with t=0, $C_y = C_y^0$, $R = R^0$, and $m = m^0$. Given the trajectory of each vehicle and the corresponding reachability in R, we extend Algorithm 1 (i.e., the Hungarian Algorithm for the QAP) in order to consider the capacity volume of Y in the bipartite matching to X. Basically, the parking place y with a vacant slot can be allocated to any waiting vehicle i and makes a record $m(i, y) = 1$ in our system. Under our capacity constraint, the status of a vehicle $x \in X$ or a place $y \in Y$ in our assignment can be determined in Definition 1, as follows.

Definition 1. *Any $x \in X$ that has not seized the reservation is called unsaturated and it has $m(x, y) \neq 1$ for every $y \in Y$. Any $y \in Y$ still available for allocation is called unsaturated and it has $C_Y - \sum_{x \in X} m(x, y) > 0$, simply $^@ y > 0$.*

When the capacity allows, a place may appear in multiple pairs of matching. This relation map is maintained in our m-table, where we also implement the shuffle and merging processes (i.e., phase 4) of Algorithm 1. The m-table and the corresponding path for the assignment shuffle are defined in Definitions 2 and 3, as follows.

Definition 2. *The tree with a root $u \in X$ is called alternating tree (or simply called E-tree) and is denoted by E_u^* when each edge $\{x, y\}$ (or $\{y, x\}$) has $m(x, y) \geq 0$.*

Definition 3. *The path existing in E_u^* is called augmenting path (or simply called E-path) and is denoted by E_u^v when such a path starts from u and ends with v. Along such a path, the corresponding m-value of its edge changes alternatively between 1 and 0.*

Algorithm 2. Slot allocation based on Algorithm 1.

```
REQUIRE: $X$, $Y$, $R$, and $C_y > 0$ for each $y \in Y$
ENSURE: bipartite matching $m(x, y) = 1$ for each $x \in X$ to $Y$
1) Initialization. $m(i, j)$="-" and $R(i, j) = -t$ for each vehicle $i$
   approaches a place $j$ at time $t$; otherwise, $R(i, j)$="-" as an
   unreachable status. Calculate $L(v)$ with Eq. (1).
2) Completion check. Apply phase 2 in Algorithm 1 to stop the
   algorithm. Otherwise, for any unsaturated (Definition 1) $x \in X$
   to set initial records $S = \{x\}$, $T = \{\}$, and $E_x^* = \{x\}$ (Definition 2).
3) Label update for any possible assignment at $\triangle$.
     • If $N(S) \neq T$, there exists a new $m(i, j) = 0$ according to the
       definition of $N(S)$ and $T$ in Table 1. That indicates an arrival of
       vehicle $i \in S$ at place $j \in N(S)$ at time $\triangle$. Go to phase 4 (the
       same as step 1 in phase 3 of Algorithm 1).
     • Otherwise, apply step 2 in phase 3 of Algorithm 1 (upon the
       update of $\alpha$ with Eq. (2) and $L$ with Eq. (3)). Thus, $\triangle$can be
       updated and $m$ will be reset with Eq. (4), for places not reserved
       only (i.e., $m \neq 1$), in order to consider more vehicle arrivals (i.e.,
       $m$=0).
4) Table construction for any $y \in N(S) - T$ and its $m$=0
   edge ($i$,$y$) connecting with $S$.
     • If $^@ y > 0$ (with Definition 1), alt er $m$=1 and 0 along the E-
       path $E_x{}^i$ based on Definition 3. Set $m(i, y) = 1$ to add this
       assignment in $E_x^*$ and $E^y{}_x$, and then go to phase 2.
     • Otherwise, $y$ was reserved ($\exists\ m(z, y) = 1$). Set $S = S \cup \{z\}$
       and $T = T \cup \{y\}$. Then, go to phase 3.
```

The details of our extension can be seen in Algorithm 2. Unlike aiming to the traditional $n \times n$ QAP, this new assignment scheme can find the best locations $\subseteq Y$ for scheduling all vehicles X to minimize the total cruising/waiting time, as we will prove in the following theorem. The key is to implement the corresponding shuffle and merging processes with m-table. Note that m=0 indicates a possible matching (see Eq. (4)) and it will be converted to a real reservation by setting m=1 in phase 4. In the initialization phase, Eq. (1) is reused, but is done so in order to consider the distance from where the reservation is made to the first possible parking place in reach along the trajectory.

$$m(x, y) = \begin{cases} 0 & L(x) + L(y) = R(x, y) \\ \text{“}-\text{”} & \textit{otherwise} \end{cases} \tag{4}$$

Theorem 2 (*the optimization achieved on average delay*). *Algorithm 2 will end with a bipartite matching m so that the total cruising time can be minimized, i.e.,* $\operatorname{argmax}_m \sum_{x \in X} \sum_{y \in Y} R(x, y) m(x, y)$ *since the time is represented in* $R \leq 0$.

Proof. Algorithm 2 is derived from Algorithm 1. Its completion with a bipartite matching can be ensured as shown in Theorem 1.

Before m=1 is granted in phase 4, the capacity allowance @ is checked. Otherwise, a longer period is considered in phase 3 with the update of L and $\triangle$, in order for more vehicles to reach available slots. The progress α is the minimum at every time. Based on the Egervary theorem, the maximum weight in the resultant matching m achieves the minimum size of L, which is derived from R. According to the definition of non-positive values in R ($\leq$0), we have the minimum of the total elapsed time. □.

Fig. 2 demonstrates step-by-step how the minimal total cruising time in Fig. 1 (e) can be achieved with Alg. 2. In this sample, we have the capacity $C(a, b, c, d, e) = (1, 1, 1, 1, 1)$ as a QAP. We also use this process to explain the shuffle and the merging of E-paths in Algorithm 2, as an extension from the Hungarian algorithm in Algorithm 1. To simplify the discussion, we use a unit road segment between any two adjacent places.

Scenario 1. Data preparation in the **initialization** phase. At step 1 in Fig. 2, not only is the arrival time interpreted in the cost table R, but also is the reachability. With the implementation of our maximum weight matching (e.g., x ray,), the non-positive R-value and the corresponding m-value is initiated (as the same in Algorithm 1). As indicated in Fig. 1 (c), vehicle 1 will take the route $b \to c \to d$ and has the arrival time: 0, 1, and 2, respectively. We have $R(1, b) = 0$, $R(1, c) = -1$, and $R(1, d) = -2$. Because a and e are not reachable, we have $R(1, a) = R(1, e)$="-".

Scenario 2. Start of a new matching in the **check** phase. At the beginning of the entire process (step 2 in Fig. 2) or when a bipartite matching is accomplished for S-set $\subset X$ (e.g., step 5 in Fig. 2), a new matching is initiated from an unsaturated vehicle (by Definition 1) as the root of an E-tree. When all vehicles are assigned, the entire procedure converges here.

Scenario 3. Seizing parking opportunities in the **label update** phase. For instance, at step 3 in Fig. 2, α can be calculated. Then, the succeeding arrival(s) with the minimum delay can be determined. With the calculation of L, we can catch the arrivals of those vehicles $\in S$ in m-table with Eq. (4). At $\alpha = 0$, we have $m(1, b) = m(2, b) = m(3, a) = m(4, a) = m(5, a) = 0$, implying where these vehicles start the cruising. This indicates a fair chance for each vehicle to seize a parking opportunity: 1 and 2 at place b (i.e., $N(\{1\}) = N(\{2\}) = \{b\}$) and 3, 4, and 5 at place a (i.e., $N(\{3\}) = N(\{4\}) = N(\{5\}) = \{a\}$). See the dashed lines in the bipartite matching.

Scenario 4. Direct assignment without shuffle in the **table construction** phase. For instance, at steps 4, 9, and 32 in Fig. 2, the root x will seize the opportunity and be assigned with an unsaturated place under the capacity constraint. It is the simplest of all three scenarios in this phase 4 of Algorithm 2. In step 4 in Fig. 2, we can locate a new place $y = b \in N(S) - T$ without being considered yet, i.e., $^@ y > 0$. We can make the assignment $m(x, y) = m(1, b) = 1$. See the m-table update in interpreting the arrow line in E_1^* and the thick line of

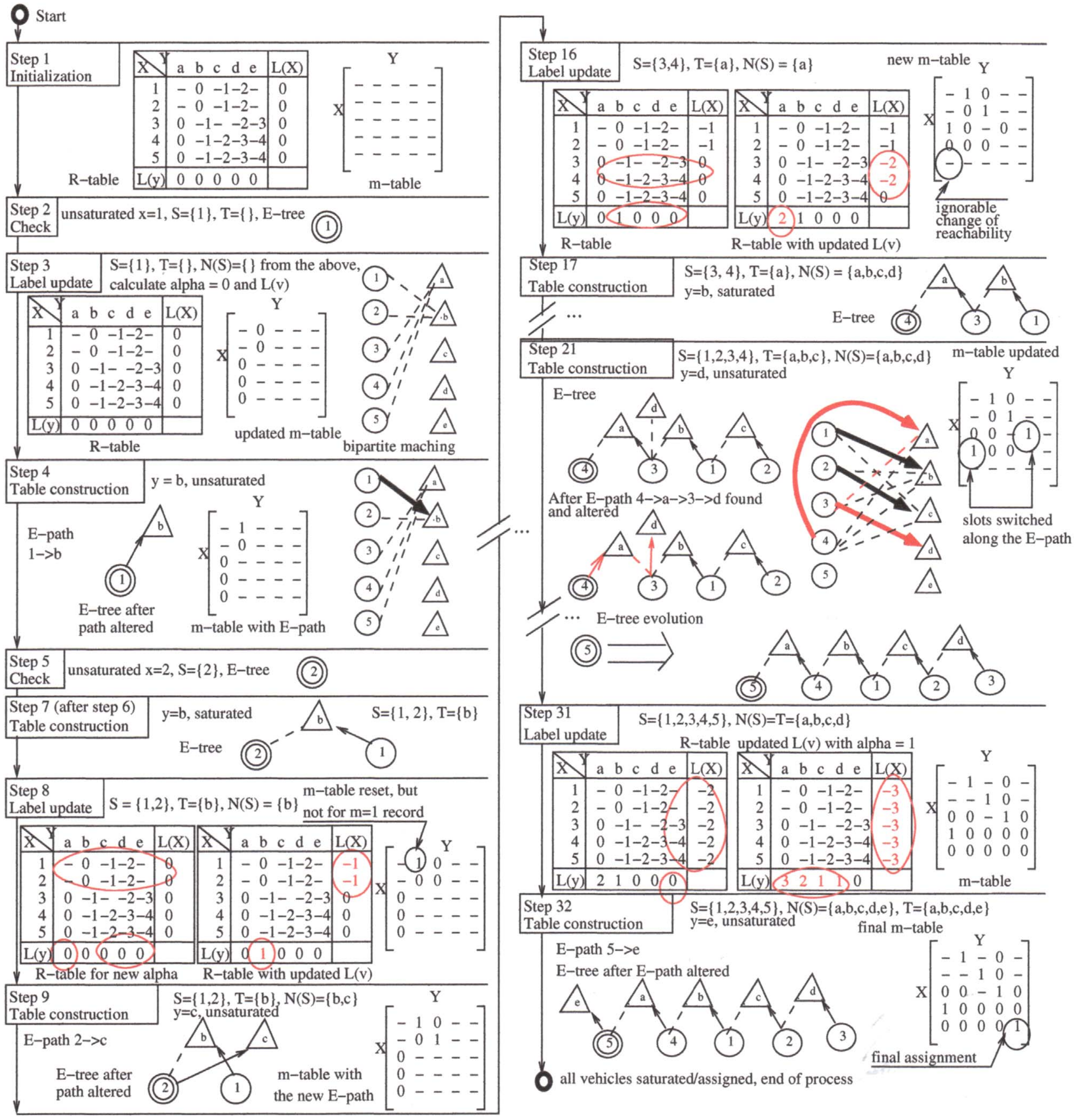

Start
Step 1 Initialization
R–table
m–table
Step 2 Check
unsaturated x=1, S={1}, T={}, E–tree
Step 3 Label update
S={1}, T={}, N(S)={} from the above, calculate alpha = 0 and L(v)
updated m–table
bipartite maching
Step 4 Table construction
y = b, unsaturated
E–path 1–>b
E–tree after path altered
m–table with E–path
Step 5 Check
unsaturated x=2, S={2}, E–tree
Step 7 (after step 6) Table construction
y=b, saturated
S={1, 2}, T={b}
Step 8 Label update
S = {1,2}, T={b}, N(S) = {b}
m–table reset, but not for m=1 record
R–table for new alpha
R–table with updated L(v)
Step 9 Table construction
S={1,2}, T={b}, N(S)={b,c} y=c, unsaturated
E–path 2–>c
m–table with the new E–path
Step 16 Label update
S={3,4}, T={a}, N(S) = {a}
new m–table
R–table with updated L(v)
ignorable change of reachability
Step 17 Table construction
S={3, 4}, T={a}, N(S) = {a,b,c,d} y=b, saturated
Step 21 Table construction
S={1,2,3,4}, T={a,b,c}, N(S)={a,b,c,d} y=d, unsaturated
m–table updated
After E–path 4–>a–>3–>d found and altered
slots switched along the E–path
E–tree evolution
Step 31 Label update
S={1,2,3,4,5}, N(S)=T={a,b,c,d}
R–table updated L(v) with alpha = 1
Step 32 Table construction
S={1,2,3,4,5}, N(S)={a,b,c,d,e}, T={a,b,c,d,e} y=e, unsaturated
final m–table
E–path 5–>e
E–tree after E–path altered
final assignment
all vehicles saturated/assigned, end of process

Fig. 2. Process with Algorithm 2 to achieve the optimization in Fig. 1 (e) where some important intermediate result at each step is highlighted in red. Note that some steps are omitted due to the limit of space. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the bipartite matching.

Scenario 5. Merging assignments in the **table construction** . For instance, at steps 7 and 17 in Fig. 2, an existing assignment will merge into the current consideration on S and T. This is a preparation in order to find a connecting path for alternating the assignments later. In step 7, $N(S) = N(\{2\}) = \{b\}$. However, $m(1, b) = 1$ as derived in step 4 early. Place b (C_b=1) does not have enough space for both vehicles 1 and 2 (i.e., $^{@}$ $b = 0$). Therefore, we merge the assignment $m(1, b) = 1$ to E_2^* (interpreted by S, T, and m) and will have $S = \{1, 2\}$ and $T = \{b\}$. This is fulfilled in the saturated case of phase 4 in Algorithm 2.

Scenario 6. Expanding $\triangle$ in the **label update** phase for more parking opportunities. For instance, at steps 8 and 16 in Fig. 2, after $N(S) = T$ is confirmed, α is calculated as the minimum time for vehicles $\in S$ to reach any place that has not been considered yet, i.e., $\in Y - T$. After that, L is updated to include both existing and new parking opportunities of vehicles $\in S$. See the updated m-table with Eq. (4) in step 8 in Fig. 2. Those records involved in this update are highlighted in red circles. Note that those m=1 records do not need any change. Only those m=0 records may be reset to an unreachable status when the corresponding vehicle is not in the current S-set (e.g., $m(5, a)$ in step 16).

Scenario 7. Shuffle to balance the bipartite matching in the **table construction** phase. For instance, at step 21 in Fig. 2, we can find a new unsaturated place y that cannot be assigned directly to the place x (i.e., the root of E-tree). However, it can be assigned directly to a preassigned vehicle i, so that the occupied slot for vehicle i can be released to x. It is called the *shuffle* process, and such a process may involve many assignments. The sequence of slot releasing, reassigning, and releasing again is denoted by the E-path ${E_x}^i$. It will continue until we achieve a new bipartite matching by considering that place y and the root vehicle x with those existing assignments. In step 21, we assign place d, by releasing the occupancy of a from vehicle 3 and making room to reassign a to vehicle 4. The resultant assignments after this shuffle are highlighted in red in the bipartite graph and with circles in the m-table.

The above process can easily be extended to the situation when a place y is capable of having multiple slots (i.e., $C_y > 1$). In the following, we prove the bound of the computational cost of our solution and the cruising/waiting time of each individual vehicle while it plays a local role in achieving the global optimization.

Theorem 3 (*the bound of computational overhead*). *The overall complexity of Algorithm 2, $T^2(n)$, is $O(n^4)$, where $n = |Y|$.*

Proof. We assume that the capacity of each place C_y is limited by a constant threshold C. So searching all available slots has the same complexity as searching all places $\in Y$. Moreover, for a complete matching, all vehicles will become saturated at the end. So we have $|X| \le C \times |Y|$. For each vehicle in phase 2, we have a loop to stop at phase 4 until the E-path can be constituted. The place matching will check every place in Y for an unsaturated candidate, while the E-path can be implemented with König's graph theorem. The complexity in tree construction can be controlled within $O(|Y|^2)$. The expanding of $\triangle$is no more complicated than $O(|Y|^2)$ because the size of S is smaller than $C \times |Y|$. Such an expanding is executed only when no unsaturated place is available. Therefore, each iteration of this matching loop has a complexity $O(|Y|^3)$ and then the statement is proven. □.

Corollary 1. $T^1(n) = T^2(n)$.

Proof. The result is obvious based on Theorems 1 and 3. □.

Corollary 1 proves that Algorithms. 1 and 2 have the equivalent complexity. Note that $|Y| < < |X|$. The complexity of our MDP solution in Algorithm 2 can be controlled with the size of garages, rather than that large amount of vehicles. That is, Algorithm 2 is a practical solution.

Corollary 2 (*the upper bound of cruising/waiting as the worst case*). *While achieving the global optimization in total cruising time, the upper bound of cruising time for any vehicle $x \in X$ in Algorithm 2 is $r = \max_{i\in X} r_i$, where r_i is the cycle of every vehicle i along its loop.*

Proof. We have assumed that there is sufficient parking vacancy. Thus, there exists a solution to match all the vehicles to their places before everyone completes a cycle in its cruising. However, such an assumption does not provide the assignment solution and cannot guarantee its optimization. In Algorithm 2, the global optimization can be achieved with Theorem 2. Moreover, any vehicle x can reach its farthest place in time r_x along the trajectory loop. Otherwise, some loops are isolated and the problem can be reconsidered as an individual in each connected graph. Therefore, the upper bound of cruising/waiting time is $r = \max_{i\in X} r_i$. □.

The arrangement in Fig. 1 (f) can also be achieved with Algorithm 2, from $C_b = C_c = 1$ at time t=0. In the next section, we will demonstrate how to schedule parking slots after considering the vacancy growth in the time scale. As a result, the assumption of sufficient vacancy at the scheduling moment can be released. In Fig. 1 (g), when only one slot is available ($C^0(a, b, c, d, e) = (0, 1, 0, 0, 0)$) at time t=0, the proposed scheme can guide both vehicles 1 and 3 to reach their targets within a time bound.

6.2. Solution with the vacancy growing along the time scale

The above solution requires the sufficient capacity to consume all parking requests, i.e., $|X| \le \sum_{y\in Y} C_y$. It cannot completely support our hot-spot parking when the demand exceeds the supply. We need to utilize the capacity growth along the time scale, which can be induced by the enforced leaving under our parking restriction or other facts. Here, we present the complete solution for our maximum-weight matching m^t in Algorithm 3.

The unsaturated status of vehicle or place can be defined as follows in Definition 4, with the consideration of capacity against occupancy along the time scale.

Definition 4. *Any $x \in X$ is called unsaturated when $m^t(x, y) \neq 1$ for every $0 \le t \le \triangle$ and $y \in Y$. Any $y \in Y$ is called unsaturated at time t when $C_y^t - \sum_{x\in X} \sum_{0\le i\le t} m^i(x, y) > 0$, simply $y^@ > 0$. Especially, we consider $y^@ > 0$ at time $\triangle$.*

The implementation is derived from Algorithm 2, with the extension of R and m to R^t and m^t respectively. For the assignment shuffle at different time $0 \le t \le \triangle$, the E-tree will consider all records in the past. Respectively, each m^t must be stored. Thus, the E-tree is built on the union $\bigcup_{0\le t\le\triangle} m^t$ and the corresponding E-path can be defined as the follows in Definition 5.

Definition 5. *The path existing in E_u^* (Definition 2) with the m-weight on each edge is called augmenting path (or simply called E-path) and is denoted by E_u^v (Definition 3) when such a path starts from u and ends with v. Along this path, the corresponding m-value of its edge $\{x, y^t\}$ or $\{y^t, x\}$ (i.e., $m^t(x, y)$ respectively) changes alternatively between 1 and 0.*

Considering a valid occupancy, each C_y^t will trigger a recalculation of R^t. R will be updated with the most recent R^t ($t \le \triangle$) where the $\triangle$is derived from the update of $L(X)$ with Eq. (3) and has the progress of α with Eq. (2). The use of Eq. (5) is to obtain the initial result in our extension, as the use of Eq. (1) in Algorithms 1 and 2. Eq. (6) resets L within the current consideration S for each update of R^t. The rest will be treated as the same as applying Algorithm 2 in an extended R-table with the consideration of capacity change in ${C_y}^t$.

$$L(v) = \begin{cases} \max_{y\in Y, 0\le t\le\nabla} R^t(v, y), & v \in X \\ 0, & v \in Y \end{cases} \tag{5}$$

$$L(v) = \begin{cases} \max_{y\in Y, 0\le t\le\nabla} R^t(v, y), & v \in S \\ 0, & v \in Y \end{cases} \tag{6}$$

Algorithm 3. Slot allocation extended from Algorithm 2 to consider the capacity growth in the time scale.

```
REQUIRE: Global time $t$, $Y$ in a indoor parking system where
$C_y^t < C_y^{t+\delta}$ for each $y \in Y$ during any period $[t, t+\delta]$, $X$, and their
cyclic routes $R^t$ after each possible ${C_y}^t$ appears.
ENSURE: saturated matching $m^t(x, y) = 1$ for each $x \in X$ to $Y$.
1) Initialization. Apply phase 1 of Algorithm 2 for $m = m^0$ and
   $R = R^0$. Calculate $\nabla$and $L$ (with Eq. (5)). Then, update $\triangle$.
2) Completion check. Set $t$=0. Apply phase 2 in Algorithm 2 to
   initiate $S$, $T$ and $E_x^*$ for an unsaturated vehicle $x \in X$ (defined in
   Definition 4); otherwise, stop the algorithm.
3) Label update for any possible assignment at $\triangle$. Repeat
   the following process until $N(S) \neq T$, then go to the next
   phase.
     • Calculate $\alpha$ with Eq. (2).
     • When $C_y^k > 0$ ($k \le \alpha + \triangle$) exists, find the minimal $k$. For
   each $x \in S$ and its $R_x = R_x^t$ where $t < k$, set $R_x = R_x^k$. Then,
   calculate $\nabla$, $L$ (with Eq. (6)), and $\triangle$.
     • Apply the label update phase of Algorithm 2 and reset each
   parking opportunity in $m$ with Eq. (4) when $m \neq 1$.
4) Table construction for any $y \in N(S) - T$ and its edge
   $m(i, y) = 0$ where $i \in S$. The same as the phase 4 in Algorithm
   2, with the saturated/unsaturated status of $y$ at time $\triangle$ defined
   in Definition 4 and the E-tree $E_u^*$ (or E-path $E_u^v$) defined in
```

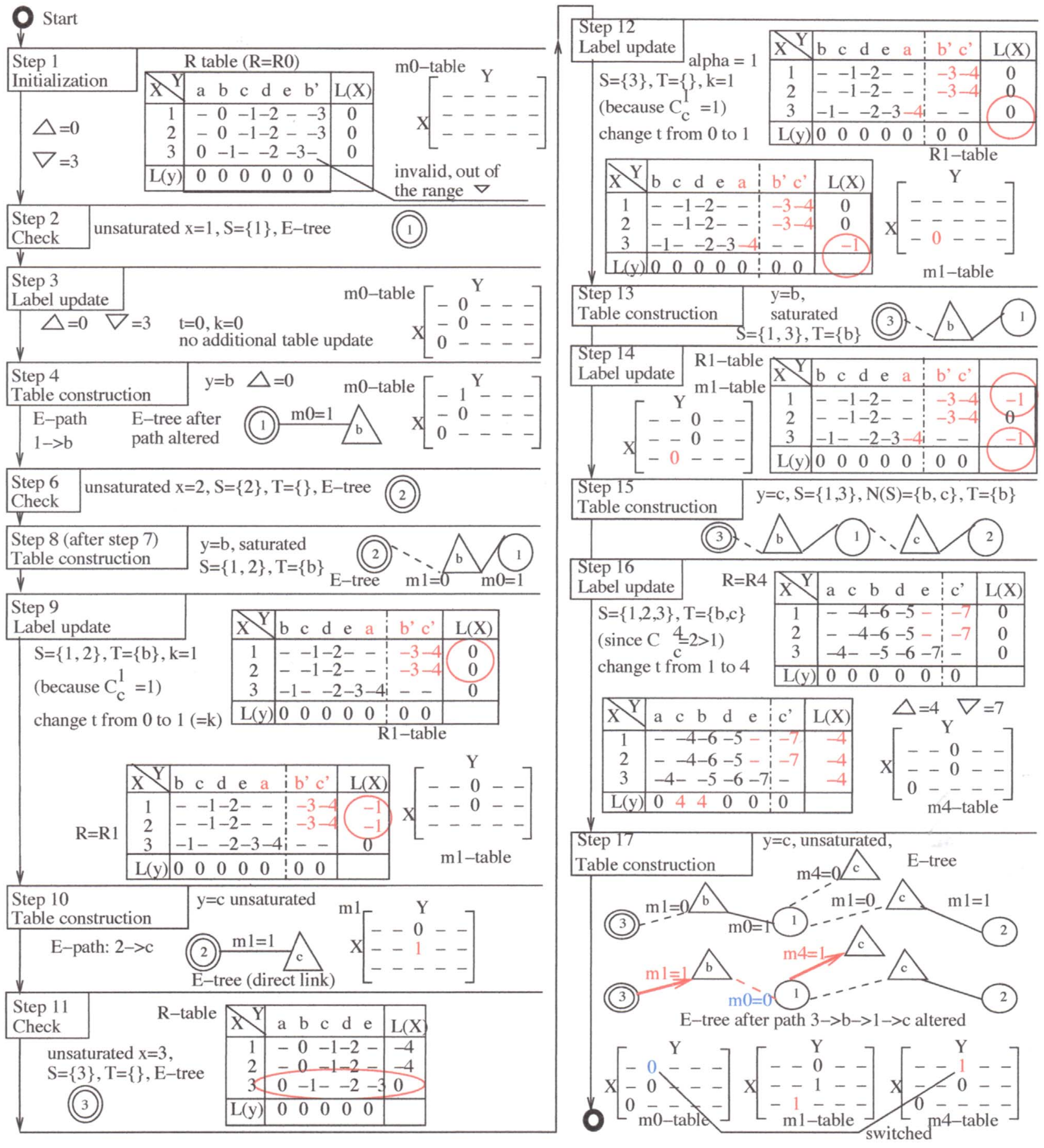


Fig. 3. Process with Algorithm 3 to achieve the optimization in Fig. 1 (g), where initially $C(a, b, c, d, e) = (0, 1, 0, 0, 0)$ and the capacity changes ($C_c^1 = 1$ and $C_c^4 = 2$); that is, a solution for the starvation problem in current shortest-path-based and assignment-based schemes of vehicle dispatching.

Definition 5.

In the following Lemma 1, we prove that capacity check at time $\triangle$ will be effective for the entire matching process. The result also ensures the feasibility of assignment shuffle between different times. Then, in Theorem 4, we prove the optimization at the global view level achieved by Algorithm 3. After that, a bounded reservation service can be ensured, in terms of the cruising/waiting time proven in Theorem 5. We also show that the computational overhead of Algorithm 3 can be controlled in an acceptable range, in terms of the limited information collected during a certain long period (i.e., $[0..(\gamma + 2r)]$ in Theorem 5) and the time complexity (in Corollary 3).

Lemma 1. *The capacity constraint @ applied on place y at time $\triangle$can ensure that no occupied slot can be included in m at any time and then be double-booked.*

Proof. Assume we have C_y^{t1} and C_y^{t2}. Without loss of generality, $t1 < t2$ and $C_y^{t1} \le C_y^{t2}$. We assume there is no other C_y^t that $t1 < t < t2$. For any new m=1 record to occupy a slot in $C_y^{t2} - C_y^{t1}$, its capacity check on R^{t2} is enough. Meanwhile, for any m=1 shuffle (along the E-path at time $\triangle$), the capacity has been checked in the past and Algorithm 3 will rely on the use of the earliest appearance. As the $\triangle$advances, the statement holds in the entire process. □.

Theorem 4 (*the optimization on average delay*). *The bipartite matching achieved with Algorithm 3 is optimal on total elapsed time in R when each C_y^t is accurate.*

Proof. With Lemma 1, the convergence can be ensured.

Derived from Algorithm 2, the pace α advances in two dimensions: one is the elapsed time in R as calculated in Eq. (2), and the other is the time t for each C_y^t update, which may induce a new R-table and the corresponding elapsed time. That is, L has the minimum of the elapsed time in R. Then, similar to the proof of Theorem 2, we have this statement proven. □.

Lemma 2. *For Algorithm 3 to schedule the vacancy growing along the time scale until time γ, a saturated matching can be achieved iff* $\sum_{0\le t\le\gamma, i\in Y} C_i^t \ge |X|$ *and* $Y(\gamma) = \{y|\min_{y\in Y}\{\sum_{0\le t\le\gamma, i\in Y} C_i^t - \sum_{0\le t\le\gamma} C_y^t - |\overline{X(y,\gamma)}|\} < 0\} = \phi$ *where* $R^t(x, y)$ *is the set of vehicles that never reach y during period* $[0..\gamma]$.

Proof. The sufficiency is obvious. We assume that we have enough vacancy slots for all $|X|$ in the entire system, but we can find $y \in Y(\gamma) \ne \phi$. Therefore, we cannot have enough vacant slots (i.e., the remaining $\sum_{0\le t\le\gamma, i\in Y} C_i^t - \sum_{0\le t\le\gamma} C_y^\gamma$) for those vehicles in $\overline{X(y,\gamma)}$.

Now, we prove the necessity. Assume that there exists an unsaturated vehicle x and its R_x. If $Y = R_x$, x goes through all parking places and we have $\sum_{0\le t\le\gamma, i\in Y} C_i^t < |X|$. If $Y \supset R_x$ and $\sum_{0\le t\le\gamma, i\in Y} C_i^t \ge |X|$, we can find $y \in Y - R_x$ to meet the above condition of $Y(\gamma)$. Otherwise, there exists another vehicle that goes through all parking places, but ends with $\sum_{0\le t\le\gamma, i\in Y} C_i^t < |X|$. □.

Theorem 5 (*the fairness and starvation-freedom*). *In Algorithm 3, the upper bound of cruising/waiting time of each vehicle is* $\gamma + r$ *and will be scheduled within time period* $[0..(\gamma + 2r)]$, *where γ is the minimum provided with Lemma 2 and* $r = \max_{x\in X} r_x$ *(see definition in Table 1).*

Proof. We assume that each C_y^t is underestimated. Due to the definition of $\vartriangle$, the exclusive reservation made with Algorithm 3 can help the last vehicle to reach the vacant slot in time $\gamma + r$. γ is the time when not only the entire area has the capacity to hold all $|X|$ vehicles, but also each vehicle has at least one vacant slot appearing within reach along its trajectory (see Lemma 2). r denotes the longest cycle of vehicle along its trajectory. The proof is obvious. After $\vartriangle$extends to $\gamma + r$, union $\bigcup m^t$ is a complete graph, and a bipartite match will be achieved.

In $R^{\gamma+r}$, $\nabla \le (\gamma + r) + r$ according to the definition of ∇. Therefore, the information collection of each trajectory is limited within the interval $[0..(\gamma + 2r)]$. □.

Corollary 3. *The overall complexity of Algorithm 3,* $T^3(n)$, *is* $O(n^5)$ *where* $n = |Y|$.

Proof. Since the capacity is limited by C, the number of a valid C_y^t and the update of R^t, is limited by $C \times |Y|$. In the iteration of interior loop, the update of α and the E-path construction both introduce the time scale. Therefore, based on the proof of Theorem 3, the complexity of the interior part is $O(|Y|^4)$. As a result, the overall cost is $O(|Y|^5)$. □.

Corollary 4 (*the scalability*). *The extension from Algorithm 2–3 is under a linear-time incremental structure.*

Proof. $T^1(n) = T^2(n) = O(n^4)$ by Theorem 3 and Corollary 1. Based on Corollary 3, $T^3(n) = nT^2(n)$. □.

Fig. 3 shows a sample process of using Algorithm 3, for scheduling the first 3 vehicles in Fig. 1 (c), under the scenario in Fig. 1 (g). The schedule of vehicles 1 and 3 is to demonstrate our solution for the starvation problem in the existing shortest-path-based PGI schemes. Since vehicles 1 and 2 adopts the same routine, the schedule of vehicles 2 and 3 here is to demonstrate our solution discussed in early Section 3 with the consideration of capacity growth along the time scale. This distinguishes our MDP from existing assignment-based scheme of vehicle dispatching. Initially we have $C(a, b, c, d, e) = (0, 1, 0, 0, 0)$ and only one vehicle is allowed to confirm the parking at t=0. After that, we have $C_c^1 = 1$ and $C_c^4 = 2$.

In step 1, vehicle 3 has the longest cycle $r = r_3 = 3$ so that $\nabla = r_3 = 3$. Then, we set $R = R^0$ and m. We add succeeding trajectories of vehicles 1 and 2 to R in order to complete the table R with the period ∇(i.e., $[0..3]$). After that, we set L and get $\vartriangle = 0$. From step 2–4, vehicle 1 is matched with the unsaturated place b, forcing the other two to wait. See the competition among vehicles 1 and 2 in step 8 (i.e., $N(S) = N(\{1, 2\}) = \{b\} = T$). In step 9, before we consider the arrivals at $t = \vartriangle + \alpha = 1$, $R = R^1$ due to a new slot available at t=1 (C_c^1). $R(3, a)$ is the last stop in the cycle from t=1 and triggers an update of $\nabla = 4$. After that, m-table is reset and these arrivals ($R(1, c)$ and $R(2, c)$) can be considered within the period $[1..4]$. Finally, in step 10, we assign place c to vehicle 2. In step 11, $R = R^0$ for the only vehicle unassigned. R-table is soon updated to R^1 in the next step. The reservation conflict is observed while the existing assignment $(1, b) = 1$ at m^0 blocks the possible assignment $(3, b) = 0$ in m^1. Then, in step 16, $R = R^4$ due to the existence of $C_c^4 = 2$. This triggers the assignment switch between vehicles 1 and 3 at place b in step 17, as we expected in Fig. 1 (g). Note that Algorithm 3 requires to check the complete m-tables along the time scale, though each update is made on a snapshot in a specified m^t.

Therefore, both starvation problems discussed early in Fig. 1 (f) are solved in our MDP solution.

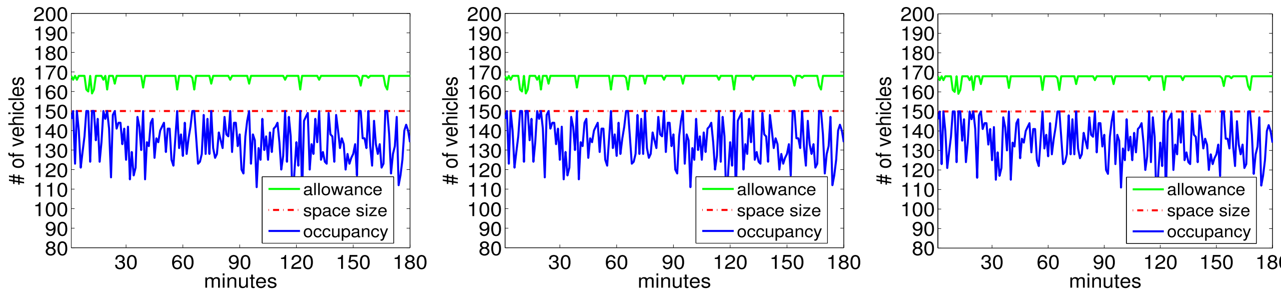
6.3. Service implementation

Algorithm 3 can be applied directly to a real road system when the cost between two adjacent parking places is various for every vehicle and represented with a decimal number in R. For any reserved vehicle to reach its slot within the expected bound, the reservation cannot be depleted. Otherwise, a starvation may incur. For every vehicle comes after an assignment is fulfilled, it can be scheduled with Algorithm 2 immediately if there exists any vacant slot in the network. This kind of service adopts the FIFO policy. Otherwise, our allocation center can wait until another unsaturated vehicle x approaches very close to an unsaturated place y. Then all unsaturated vehicles that have submitted their requests will be scheduled with Algorithm 3, to those potential places that will create vacancy along the time scale. The time between two consecutive allocations is called the schedule interval (Geng and Cassandras, 2013) and can be used to avoid the computation overhead that is triggered by every incoming request. Therefore, a seamless service can be provided and every parking request can be satisfied. There might be room to improve the scheduling of two consecutive MDP assignments between an interval time period. However, the need for a separated MDP assignment is due to the lager of request sending. Such deviation to the theoretical optimization is out of the technical scope and the corresponding solution for the drivers to obtain a better reservation (e.g., Polak and Axhausen, 1989) is omitted.

7. Simulation

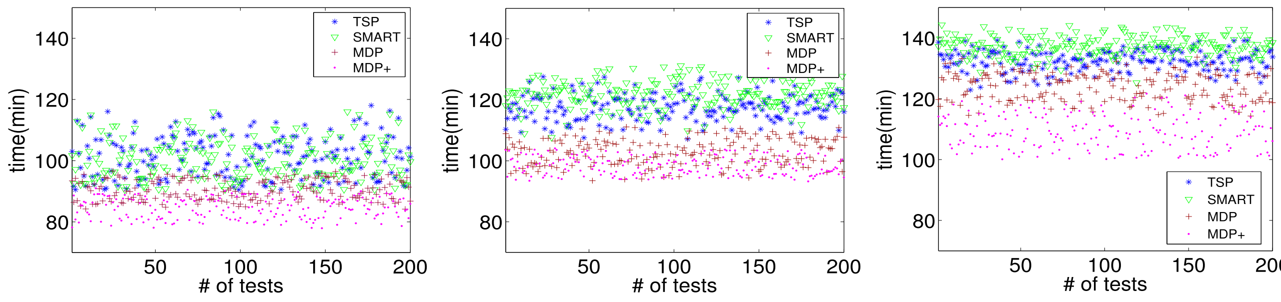
We set up our simulation based on the real scenario around the West Lake. The street map is derived from OpenStreetMap (Haklay and Weber, 2008). The trace data of each vehicle is generated by the simulation SUMO (Behrisch et al., 2011), based on the traffic information fetched via all media channels such as Yang (2014) and our road tests. Note that this simulation work is not to show whether we can cover all schedules in the entire lake area. Indeed, it is a smallest field environment to cover all possible delay problems in other existing work, including those discussed in both Sections 3 and 6. This is used to prove the delay improvement under a real traffic mode by our MDP solution (also MDP+).

We chose 5 major parking facilities for the attractions around the lake as shown in Fig. 1. To avoid the effect of bottleneck, we assume that each place has a uniformed size C=150 (in red dash-dot line in



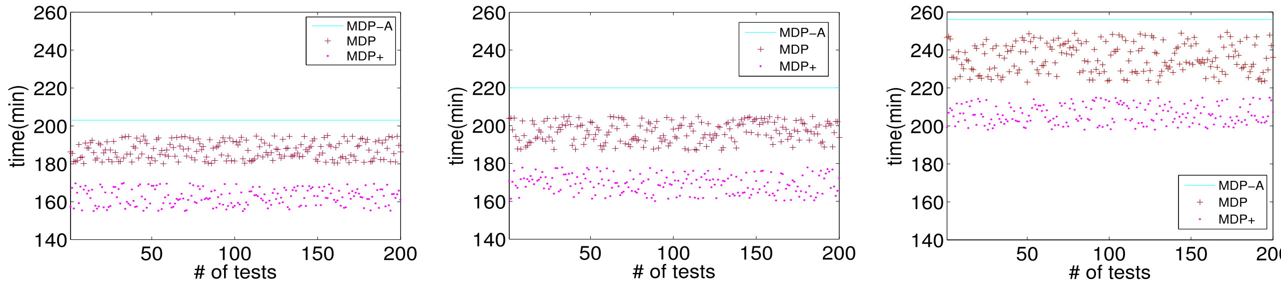
(a) daily traffic (b) weekend traffic (c) Golden Week Holiday

Fig. 4. Size, occupancy, and capacity of a parking place in different traffic models.



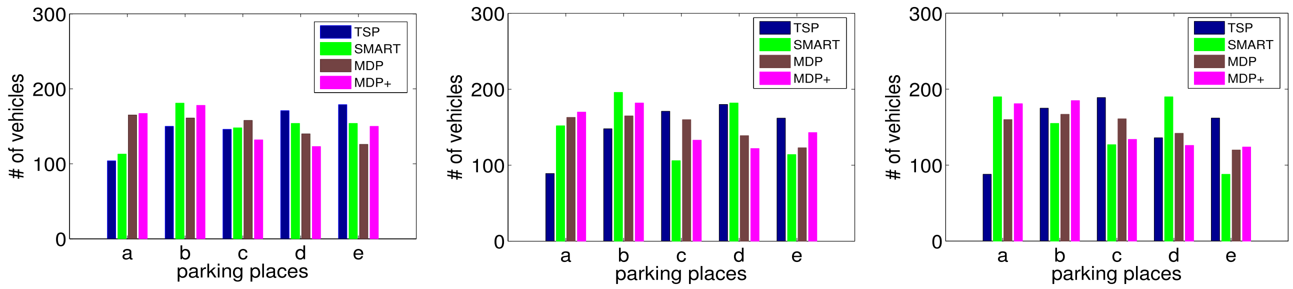
(a) daily traffic (b) weekend traffic (c) Golden Week Holiday

Fig. 5. Average cruising time of each vehicle (i.e., time to stay in the traffic) in each test case.



(a) daily traffic (b) weekend traffic (c) Golden Week Holiday

Fig. 6. Maximum cruising time in MDP and MDP+, compared with the upper bound in our analysis.



(a) daily traffic (b) weekend traffic (c) Golden Week Holiday

Fig. 7. Distribution of vehicle assignments (on average).

Fig. 4). The occupancy of daily parking, weekend parking, and the one during the Golden Week Holiday is shown in Fig. 4 in blue lines, under a continuous surveillance of 3 h. Each represents the situation of low, medium, and high traffic volume, respectively. When the vehicles are allowed to dwell around the reserved place due to the cost for driving to the next available place, more reservations are allowed than the capacity of each place (denoted by “allowance” in green line in Fig. 4).

This data of occupancy also implies the amount of parking needs at any moment along the time scale. We randomly select a scenario from Fig. 4, and then apply different PGI schemes to schedule those vehicles coming to reserve the vacant slots. We test TSP (Ayala et al., 2012), SMART (Geng and Cassandras, 2013), our MDP, and its extension MDP+ with the dwelling around the reserved place. SMART allows the user to select the closest place, while TSP simply follows the predetermined trajectory to the next available one. In SMART, MDP and MDP+, each driver will predetermine a routine trajectory according to the travel plan and then submit the request with such path information. The place from where the request is received by the center will be considered as the starting point in the scheduling, and can be different according to the timing of submission. Along such a trajectory, the arrival at each parking place is determined by the GPS in different modes of traffic volume: daily parking low, weekend parking medium, and Golden Week Holiday high, with a considerable change of +/ − 25% on the time/speed that is estimated with Asif et al. (2014), Ganti et al. (2014) in our MDP calculation. That is due to the impact of road traffic or congestion in real time. Note that, in existing schemes TSP and SMART, the starvation problem cannot be solved. In this paper, we show the results from the first 200 times of scheduling, which are consistent with all of our results.

In Fig. 5, we show the average cruising time needed for vehicles to obtain their parking opportunities. Fig. 6 shows the worst case with our analysis on its bound in Theorem 5, which is denoted by MDP-A. In order to facilitate the calculation, the simulation adopts the commonly used 3-h parking limit to ensure the time $\gamma < 180$ minutes in Lemma 2. Fig. 7 shows the distribution of slot allocation.

Our observations are summarized in fourfold: **(1)** From Fig. 5, the global optimization on total cruising time can be seen obviously via the average cruising time in MDP and MDP+, compared with those existing PGIs. MDP, TSP, and SMART are listed as the second, third and fourth. Note that we focus on the parking in an extremely crowded area, not the one in a small town. Without any guidance, the cruising may need hours even in the weekdays (with low traffic volume), and becomes worse in the weekends (with medium traffic volume) and the holiday seasons (with high traffic volume). **(2)** Fig. 5 also shows that the average cruising of each individual vehicle under MDP or MDP+ can outperform those in the existing PGIs. When the parking demands exceed the supply, our results show that the simple rule in TSP outperforms the flexibility in SMART. Our MDP always outperforms TSP and SMART, while MDP+ sacrifices the driving speed (slowly cruising around and waiting in the local area) to reduce the cruising time. The time reduction in MDP+, compared with MDP, convinces us about the fact that roughly knowing how long to wait will favor the driver. **(3)** Fig. 6 shows the effectiveness of our performance bound in MDP as the analysis provided in Theorem 5. The worst case on that bound is even better than the average achieved in the existing PGIs, showing the substantial improvement of our MDP. This bound can be lifted up when the number of vehicles increases in the denser traffic condition, and can also be extended to MDP+ after the corresponding data in estimation is used. **(4)** Fig. 7 shows the distribution of reservations. No place in our MDP and MDP+ will have the occupancy exceeding the maximum one in either TSP or SMART. Our new schemes have the capability to obtain the parking opportunity in the entire network, without relying on any specified place. This proves the fairness in our MDP and MDP+ assignments. Moreover, MDP and MDP+ are distance-relevant schemes and will prefer those places close to the starting point of each vehicle. The results show that many drivers living in center city will enter the system from place *b*. For people from suburban areas and even nearby cities, place *a* is more convenient because it is close to the highway exit.

8. Conclusion and future work

In this paper, a new PGI, denoted by MDP, has been proposed, in order to mitigate the impact of the parking hassle of delay (in both the average and the worst cases). We provide a spatiotemporal assignment, in order to take advantage of the vacancy that grows along the time scale when the demands exceeds the supply. The unique directive is to solve the aforementioned starvation problem in other PGI schemes or similar vehicle dispatching. The contribution is to reduce delay without increasing the facility supply. Both analytical and experimental results demonstrate that our approach can achieve a bounded service, in terms of vehicle cruising time and the overhead cost of information collection and computation. Moreover, we study the extension by trading in the local waiting when the driver knows how soon the vacancy becomes available. The corresponding assignment is denoted by MDP+. After that, a full service can be provided for scheduling every parking request.

In our future work, we will consider the capacity decadence when both assisted and non-assisted drivers co-exist in the parking field. We will st udy the tradeoff between the global optimization and the greedy approximation algorithm, so that even more practical solutions can be achieved. We also expect to apply this spatiotemporal assignment scheme to other resource shortage problems (e.g., Akhtar et al., 2016; Khan et al., 2016), while a global optimization is desired.

Acknowledgement

This research was supported by the National Science Foundation grants CCF 1028167, CNS 1065444, CNS 1138963, ECCS 1128209, and ECCS 1231461. This work is also supported by the Natural Science Foundation of China (by grants 61601157 and 61272539), the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry), and the Chinese Scholarship Council (by grant 201208330096).

References

Abdullah, K., Kamis, N., Azahar, N., Shariff, S., Musa, Z., 2012. Optimization of the parking spaces: A case study of Dataran Mawar, UiTM Shah Alam. In: Proceedings of IEEE CHUSER.
Akhtar, F., Rehmani, M., Reisslein, M., 2016. White space: definitional perspectives and their role in exploiting spectrum opportunities. Telecommun. Policy 40 (4), 319–331.
Alfonsetti, E., Weeraddana, P., Fischione, C., 2015. Min-max fair car-parking slot assignment. In: Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM).
Arif, S., Olariu, S., Wang, J., Yan, G., Yang, W., Khalil, I., 2012. Datacenter at the airport: reasoning about time-dependent parking lot occupancy. IEEE Trans. Parallel Distrib. Syst. 23 (11), 2067–2080.
Asif, M., Dauwels, J., Goh, C., Oran, A., Fathi, E., Xu, M., Dhanya, M., Mitrovic, N., Jailet, P., 2014. Spatio and temporal patterns in large-scale traffic speed prediction. IEEE Trans. Intell. Transprotation Syst. 15 (2), 797–804.
Ayala, D., Wofson, O., Xu, B., Dasgupta, B., Lin, J., 2012. Parking in competitive settings: A gravitational approach. Proceedings of IEEE MDM.
Ayala, D., Wofson, O., Xu, B., Dasgupta, B., Lin, J., 2012. Pricing of parking for congestion reduction. Proceedings of SIGSPATIAL.
Baroffio, L., Bondi, L., Cesana, M., Rdondi, A., Tagliasacchi, M., 2015. A visual sensor network for parking lot occupancy detection in smart cities. Proceedings of the IEEE World Forum on Internet of Things (WF-loT).
Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D., 2011. SUMO-simulation of urban mobility: An overview, Proceedings of SIMUL.
Benenson, I., Martens, K., Birr, S., 2008. Parkagent: an agent-based model of parking in the city. Comput., Environ. Urban Syst. 32 (November (6)), 431–439.
Blog, G.-C.-T. Does beijing really need 2.5 million parking lots? Sustainable Transport in China, document is available at ⟨http://sustainabletransport.org/does-beijing-really-need-2-5-million-parking-lots/⟩.
Bondy, J., Murty, U., 1976. Graph Theory with Applications. 1st ed. Elsevier Science Publishing Co., Inc.,
Choeychuen, K., 2013. Automatic parking lot mapping for available parking space detection. Proceedings of IEEE KST.
Espanol, F., 2011. Ancient chinese cultural landscape, the west lake of hangzhou, inscribed on UNESCO's world heritage list. UNESCO, June.
Ganti, R., Srivatsa, M., Abdelzaher, T., 2014. On limites of travel time predictions: Insights from a new york city case study. Proceedings of the IEEE ICDCS.
Gao, G., Xiao, M., Zhao, Z., 2016. Optimal multi-taxi dispatch for mobile taxi-hailing systems. Proceedings of the 45th International Conference on Parallel Processing.
Geng, Y., Cassandras, C., 2013. New smart parking system based on resource allocation and reservations. IEEE Trans. Intell. Transp. Syst. 14 (3), 1129–1139.
Haklay, M., Weber, P., 2008. Open Street Map: User-generated street maps. In: Proceedings of IEEE PerCom.
Jin, C., Wang, L., Shu, L., Feng, Y., Xu, X., 2012. A fairness-aware smart parking scheme aided by parking lots. Proceedings of IEEE International Conference on Communications (IEEE ICC).
Khan, U., Dilshad, N., Rehmani, M., Umer, T., 2016. Fairness in cognitive radio networks: models, measurement methods, applications, and future research directions. J. Netw. Comput. Appl. 73, 12–26.
List of parking spaces in west lake scenic area, ⟨Dahangzhou.COM⟩. February 2006, document is available at ⟨http://hot.dahangzhou.com/top/tczn/2.htm⟩.
Maciejewski, M., Bichoff, J., Nagel, K., 2016. An assignment-based approach to efficient real-time city-scale taxi dispatching. IEEE Intell. Transp. Syst. 31 (1), 68–77.
Miao, F., Han, S., Lin, S., Stankovic, J., Zhang, D., Munir, S., Huang, H., He, T., Pappas, G., 2016. Taxi dispatch with real-time sensing data in metropolitan areas: a receding horizon control approach. IEEE Trans. Autom. Sci. Eng. 13 (2), 463–478.
Millikin, M., 2013. Hangzhou city begins construction of vertical parking unit for EV sharing. Green Car Congress, June.
No parking: A dearth of parking places riles the new middle class. The Economist, march 24th, 2012, document is available at ⟨http://www.economist.com/node/21551084⟩.
Pierce, G., Shop, D., 2013. SF park: pricing parking by demand. ACCESS 43, 20–28.

Polak, J., Axhausen, K., 1989. Clamp: A macrosopic simulation model for parking policy analysis. In: Proceedings of the 68th Annual Meeting of the Transportaion Research Board.
Salpietro, R., Bedogni, L., Felice, M., Bononi, L., 2015. Park here! a smart parking system based on smartphones' embedded sensors and short range communication technologies. Proceedings of the IEEE WF-loT.
Sana, B., Riadh, H., Rafaa, M., 2014. Intelligent parking management system by multi-agent approach: The case of urban area of tunis. Proceedings of International Conference on Advanced Logistics and Transport.
theexpiredmeter.com, Parking problems for fourth of July fireworks. The Expired Meter, 2010, document is available at ⟨http://theexpiredmeter.com/2010/07/parking-problems-for-fourth-of-july-fireworks/⟩.
Wan, J., Zhang, D., Zhao, S., Yang, L., Lloret, J., 2014. Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions. IEEE Commun. Mag. 52 (8), 106–113.
Wikipedia. Quadratic assignment problem, information available at ⟨http://en.wikipedia.org/wiki/Quadratic_assignment_problem⟩.
x ray. Assignment problem and hungarian algorithm, topcoder, document available at ⟨http://community.topcoder.com/community/data-science/data-science-tutorials/assignment-problem-and-hugarian-algorithm/⟩.
Yan, G., Yang, W., Rawat, D., Olariu, S., 2011. Smart parking: a secure and intelligent parking system. IEEE Intell. Transp. Syst. 3 (1), 18–30.
Yang, J., 2014. Yesterday afternoon's traffic volume is 3 times the usual as holiday makers return. Qianjiang Evening News, April.
Young, W., 1991. Parksim 1.1 users manual, Department of Civil Engineering, melbourne. Australia: Monash University.
Zheng, Y., Rajasegarar, S., Leckie, C., 2015. Parking availability prediction for sensor-enabled car parks in smart cities. IEEE ISSNIP.

Peng Liu received B.S. and M.S. in Computer Science and Technology from Hangzhou Dianzi University in 2001 and 2004, and D.Eng in Computer Science and Technology from Zhejiang University in 2007, China. Currently he is an associate professor at Hangzhou Dianzi University. His research interest include embedded system, wireless sensor networks and mobile computing.

Biao Xu is currently a graduate student at Hangzhou Dianzi University. His research includes mobile adhoc networks.

Guojun Dai received his Ph.D. in Zhejiang University in 1998. He is now a professor and the chair of Institute of Computer Applications, Hangzhou Dianzi University. His research interests include wireless sensor networks, Internet of things, embedded system, and computer visual.

Zhen Jiang received B.S., M.S., and Ph.D. from Shanghai Jiaotong University in 1992, Nanjing University in 1998, and Florida Atlantic University in 2002, respectively. Currently, he is an associate professor of the CS department at West Chester University and an adjunct professor of the CIS department at Temple University. His interests are in the area of information system development and wireless communications. He is a member of the IEEE and ACM.

Jie Wu is a Laura H. Carnell Professor in the CIS department at Temple University. Prior to joining Temple University, he was a program director at the National Science Foundation and Distinguished Professor at Florida Atlantic Univ. His current research interests include mobile computing and wireless networks, routing, cloud and green computing, network trust and security, and social network applications. He regularly published in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE TC, IEEE TSC, and JPDC. He was general chair for ACM MobiHoc'14, IEEE'13, IEEE MASS'06 and IEEE IPDPS'08 and program chair/co-chair for IEEE INFOCOM'11 and CCF CNCC'13. He is a Fellow of the IEEE and the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.