Two-stage Matching-and-scheduling Algorithm for Real-time Private Parking-sharing Programs

Pengyu Yan[a], Xiaoqiang Cai[b,c], Debing Ni[a], Feng Chu[d,e,*], and Heng He[a]

[a]*School of Management and Economics, University of Electronic Science and Technology of China,Chengdu, P.R. China*
[b]*Shenzhen Key Laboratory of IoT Intelligent Systems and Wireless Network Technology, The Chinese University of Hong Kong, Shenzhen, P.R. China*
[c]*Shenzhen Research Institute of Big Data,Guangdong, P.R. China*
[d]*School of Economics and Management, Fuzhou University, Fuzhou 350116, P.R. China*
[e]*Laboratoire IBISC, Univ Evry, Universitie Paris-Saclay, 91025, Evry, France*

Abstract

This paper studies a real-time parking-sharing program with which owners of private parking spaces can lend their parking spaces to other drivers to park when these are not in use. Compared with curbside and garage parking problems, the information of supplies and demands is randomly announced by drivers and owners respectively via a parking-sharing APP installed on their smartphones. Besides, the parking spaces made available by independent owners are usually heterogeneous in terms of their locations and available time intervals. Thus, two critical issues need to be resolved: (a) appropriately matching demands and supplies under an uncertain setting; and (b) efficiently scheduling the demands matched to avoid potential time conflicts. We propose a novel real-time reservation approach based on a rolling-horizon framework, which can assign multiple drivers to a single parking space in order to better utilize scarce parking resources. For each period, an integrated optimal matching-and-scheduling problem is formulated as a mixed integer programming model and proved to be strongly NP-hard. To fast generate a near-optimal solution to the problem, a two-stage heuristics derived from the minimum-cost flow problem is developed. The computational results validate the efficiency and effectiveness of the proposed approach. Some operational insights are also presented and discussed.

Keywords:
Parking sharing; reservation system; matching and scheduling; algorithm;

*Corresponding author
Email address: `feng.chu@univ-evry.fr` (Feng Chu)

1. Introduction

Shortage of parking resources is a significant problem in many large cities. It is reported that about 30% of vehicles in central areas of a typical city are cruising for parking and the average cruising time is 8.1 minutes per car [24]. Such inefficient transportation activities not only waste drivers' time and fuel but also cause additional traffic congestion, gas emission, and traffic accidents [13]. On the other hand, in most metropolitan areas (Beijing for instance), about 44% of private parking spaces are found to be idle during the daytime. How to best utilize the private parking spaces has emerged to be an important topic for academic research and industrial applications.

In the past decades, more and more parking-sharing programs have been launched with the support of advanced technologies (e.g., mobile-Internet, R-FID, infrastructure-to-vehicle, and vehicle-to-infrastructure communications), such as "ParkEasier" (`https://parkeasier.com/`) in Los Angeles, Boston, Chicago and other large cities of US, "JustParking" (`https://justpark.com/`) in London of UK, and "DingdingParking" (`http://dingdingtingche.com/`) in Beijing of China. These programs provide efficient alternatives for drivers who search for parking spaces nearby their destinations and meanwhile provide e-conomic compensations for owners who rent out their private parking spaces for other drivers to use. More importantly, the effective usage of private parking resources can relieve the parking pressure and reduce the traffic congestion and emissions for the whole society[24]. It is worth noting that parking-sharing programs have become especially important in many Asian metropolis, such as Beijing, Shanghai, and Hong Kong, where residential communities and workplaces are nearby each other and therefore opportunities for parking-sharing applications arise naturally[32].

The operational protocol of a parking-sharing program is illustrated in Figure 1. The program has launched an APP installed on users' smartphones. Using the APP, an owner can submit supply information to the system, including the location of his private parking space and the available time interval (e.g.,

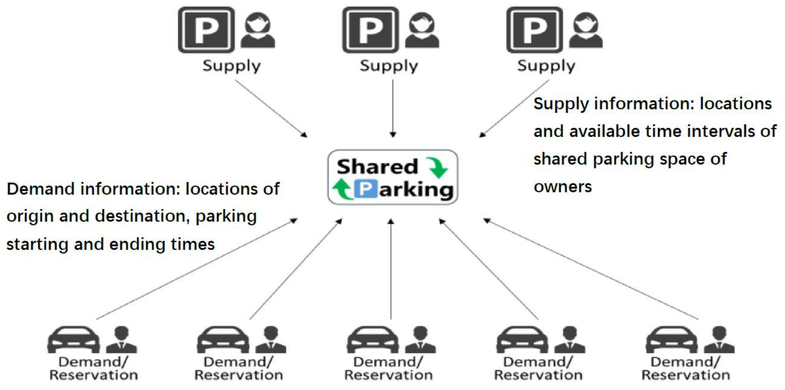

Supply
Supply
Supply
Supply information: locations and available time intervals of shared parking space of owners
Shared Parking
Demand information: locations of origin and destination, parking starting and ending times
Demand/ Reservation
Demand/ Reservation
Demand/ Reservation
Demand/ Reservation
Demand/ Reservation

Figure 1: Illustration of parking-sharing programs

from 7:00 am to 6:00 pm). A driver, on the other hand, submit her demand information to the system, including her origin, destination, and the parking starting and ending times (e.g., from 9:00 am to 5:00 pm). With a certain matching policy, for example *first-book-first-serve* rule, the system sends back a confirmed reservation to the driver who is successfully matched with a parking space. The confirmed reservation tells the driver the detailed location of the reserved parking space and the parking fee charged by the system, and the starting and ending times of using the parking space. The system also sends a message to the owner whose parking space is reserved, informing him of the driver's car information (plate number and brand), the parking income paid by the system, and the starting and ending times of using the parking space.

As illustrated in Figure 1, the parking-sharing program is a typical bilateral platform (market) connecting two distinct user groups, i.e., drivers and owners. In order to reach the economies of scale, the platform needs to attract as many drivers and owners as possible to participate in the program and to match parking demands and supplies together. Currently, to the best of our knowledge, many parking-sharing programs already launched are not as popular as expected. The reasons are multifold. Firstly, shared parking spaces supplied by owners may not be convenient to use due to the long walking time from the parking space to the final destination. Secondly, most programs allow owners to temporarily supply their shared parking spaces whenever they do not use them. The supplies, as well as the demands, may fluctuate significantly throughout a day. The program cannot know either the announcement times of drivers and owners or the exact information of the demands and supplies in advance. We recognize that individual mobility may have a regular daily trajectory [11], but there can be occasions where irregular travel needs occur (please see Appendix A.1). This paper focuses on the parking-sharing problem to address the irregular travel needs of drivers and owners. Any parking reservation system can be easily adapted by pre-treating the regular drivers and owners with the early-bird-fee rule or a long-term contract firstly, then processing the uncertain demands and supplies of irregular drivers and owners, respectively. Besides, the shared parking resources are usually heterogeneous in terms of the locations and available time intervals since they are shared by individual owners. Lastly, the current reservation systems are usually low-efficient and cannot achieve the global optimal performance of the whole system since the systems let drivers manually select and reserve their best parking spaces and even negotiate the parking fees with owners. The characteristics and problems mentioned above make the operations management of the program much more challenging than conventional curbside and garage parking problems in which parking resources are relatively fixed and deterministic. Therefore, a real-time and dynamic environment is considered in this paper and a smart and sophisticated real-time parking reservation approach involving matching and scheduling procedure is needed to developed to maximize the performance of the whole system.

The parking management problem has attracted considerable attention in recent years. Early study aims to reduce the cruising time on streets by providing drivers with real-time parking information and further guide drivers to find available parking slots on the street or in the garages [21]. Real-time information of parking space availability is often collected by sensors at entrances and exits or at individual parking slots in garages. For the on-street parking, crowdsourcing is an emerging approach to collect parking information [1, 4]. To further improve the parking experience and increase utilization of parking s-paces, some researchers have tried to develop smart parking reservation systems in which drivers can not only check the available parking spaces but also make reservations in advance [34, 33, 15, 16, 5, 25, 28]. A variety of mathematical programming models and algorithms are proposed to assign limited parking spaces to drivers with different objectives, such as the maximum revenues of a parking reservation system or the minimum drivers walking time from the assigned parking space to their desired destinations [35, 7, 10, 14, 20]. The aforementioned literature focuses on curbside and garage parking problems, where the parking spaces are usually seen as homogeneous resources with constant supplies. Also, the scheduling of drivers' parking times is not considered since drivers usually can park their cars at the reserved spaces as long as they can afford the parking fees. For a comprehensive review on curbside and garage parking problems, please refer to [13].

More and more researches are now focused on private parking-sharing programs, which assign an available private parking space to satisfy a driver's demand. Shao et al. [23] first propose a simple but efficient reservation system to maximize the profit of the e-platform or the utilization of shared parking resources. A mixed integer programming (MIP) model is established and resolved by an existing commercial software tool. They indicate that the proposed MIP approach outperforms the *first-book-first-serve* rule in their numerical experiments. Chou et al. [6] propose a reservation policy for accepting or rejecting drivers' demands to maximize revenue in a private parking-sharing system. From the perspective of economics, Xu et al. [32] design direct and indirect exchange market mechanisms with money flow to incentivize more drivers and owners to take part in the shared parking market. Xiao and Xu [30] propose a novel double VCG auction mechanism to effectively restrain both restrain drivers and owners opt out in the shared parking market. Xiao et al. [31] propose two truthful double auction mechanisms with two novel principles to respectively encourage drivers and owners to report their truth information to the parking-sharing system.

In the literature reviewed above, the parking reservation systems and mechanisms are designed under a static environment in which parking demand and supply are assumed to be known parameters. Besides, the matching models and algorithms developed can only assign a single parking space to at most one driver, which is known as an *one-to-one* matching pattern, even though more than one driver can share the same parking space when there is no overlapping in their parking times. In a practical environment, the parking reservation system should be designed to deal with uncertainties of parking demand and supply. Guo et al. [12] use a Gaussian mixture model to describe the random arrival and departure times of drivers and then develop a simulation-based optimization tool for a parking-sharing e-platform to maximize the expected profits. Recently, Wang and Wang [28] propose an innovative flexible parking reservation system to mitigate the effects of parking uncertainty from both demand and supply.

Motivated by the needs in parking-sharing applications and researches, this study proposes a real-time reservation approach to deal with a private parking-sharing problem in which demands and supplies occur randomly. The reservation system runs in a periodic operational mode, and a rolling-horizon framework is applied to roll the system forwards in a finite operational horizon. With the information on the demands and supplies collected during a period, the system may appropriately match multiple drivers with a single parking space, referred to as a *multi-to-one* matching pattern, and efficiently schedule the matched demands at the same parking space. The integrated matching-and-scheduling problem is formulated as an MIP model and proved to be strongly NP-hard. To efficiently tackle the MIP model in a real-time operational setting, a set of pre-processing constraints are first introduced to reduce the solution space. Then, an iterative two-stage heuristic algorithm is developed to fast generate a near-optimal solution. Finally, a simulation experiment setting is established based on the real data from "DingdingParking" application in a CBD of Beijing city. Experimental results of single- and multi-period scenarios show that the proposed matching-and-scheduling approach can significantly reduce the total traveling cost of all drivers and improve the utilization of parking spaces, compared with the existing one-to-one matching pattern.

The contributions of this work are highlighted as follows.

- A novel real-time parking reservation approach is proposed for a private parking-sharing program in an uncertain environment. To deal with the fluctuations in demand and supply, the system is designed in a periodic operational mode with a rolling-horizon framework. In each period, the received demands and supplies are optimally matched with a multiple-to-one matching pattern in which a parking space can be assigned to more than one driver if there is no overlapping in their parking times. Moreover, the matched demands of multiple drivers are scheduled to avoid potential time conflicts of using the same parking space.

- To tackle the integrated matching-and-scheduling problem efficiently in the real-time setting, the computational complexity of the problem is first analyzed, and then an iterative two-stage heuristic algorithm is proposed. In the first stage of the algorithm, a relaxed matching model is reduced to a network flow problem and is solved based on a revised capacitated minimum-cost flow algorithm [18]. In the second stage, a simple but efficiently heuristic rule is developed to schedule the matched demands at the same parking space.

- A simulation experiment bed is established based on the real data from "DingdingParking" application. Experimental results show that the proposed multiple-to-one matching pattern outperforms the existing one-to-one matching pattern. Compared with the commercial software tool, ILOG-CPLEX (version 12.80), the proposed iterative two-stage algorithm can fast find a near-optimal solution for large scale instances.

The rest of the paper is organized as follows. Problem description and analysis are presented in Section 2. The formulation of the problem and the proof of the computational complexity are presented in Section 3. The iterative two-stage algorithm and the computational results are presented in Sections 4 and 5, respectively. We discuss the extensions and future study in Section 6, and conclude this paper in Section 7.

2. Problem description and analysis

2.1. Periodic operations of real-time parking reservation system

We address a real-time reservation problem for a parking-sharing program. The finite operational horizon on each weekday is equally divided into $\mathbf{T}$ decision periods. The system collects parking demands and supplies announced by drivers and owners, respectively, for each period T, $T \in \{1, 2, 3, ... \mathbf{T}\}$. At the end of period T, the system provides an optimal matching-and-scheduling solution based on the collected information. Subsequently, a successfully matched driver receives a confirmation on her demand: The location of the parking space and the starting and ending times of using the space. Meanwhile, the owner, whose parking space is assigned to the driver, receives a message: The starting and ending times of his parking space being occupied and the number plate of the driver's car. Then, the successfully matched drivers and the parking spaces with zero available time are removed from the demand and supply lists, respectively. The unmatched drivers stay in the list until they are successfully matched or their demands are expired in a future period. The available time interval of some owners may be divided into several subintervals if their parking spaces are shared by multiple drivers. In such a case, we see each subinterval as a single parking space owned by a dummy owner with the same location. In the next period $T + 1$, the system generates a new matching-and-scheduling solution based on the updated lists of demands and supplies. The reservation system rolls forwards by a rolling-horizon framework.

2.2. Description of the problem

For period T, $T \in \{1, 2, 3, ... \mathbf{T}\}$, there are a list of drivers I and a list of owners J in the system, as illustrated in Figure 2. For driver i, $i \in I$,

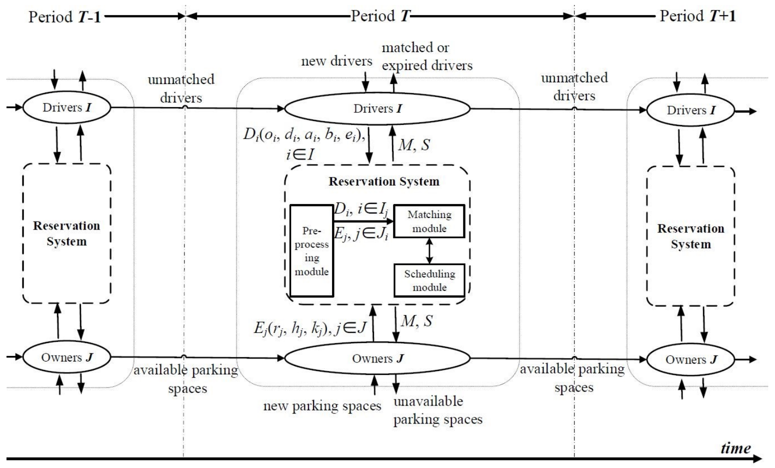


Figure 2: Periodic operations of the parking-sharing reservation system

her demand information is denoted as $D_i(o_i, d_i, a_i, b_i, e_i)$, where o_i and d_i are respectively the locations of her current origin and desired destination; a_i and b_i are respectively the earliest departure time from the origin and the requested latest arrival time at the destination; and e_i is the estimated staying duration at the destination. For a parking space $j, j \in J$, his supply information is denoted as $E_j(r_j, h_j, k_j)$, where r_j is the location of the shared parking space and $[h_j, k_j]$ denotes the available time interval of the parking space. With the collected demands and supplies, the reservation system generates a matching solution $M = \{m(i,j)|i \in I, j \in J\}$, and the corresponding scheduling scheme $S = \{(s_{i,j}, s'_{i,j})|\forall m(i,j) \in M\}$, where $m(i,j)$ indicates that driver (i.e., demand) i is matched with parking space (i.e., supply) j , and $s_{i,j}$ and $s'_{i,j}$ are the scheduled starting and ending times of driver i for parking space j, respectively. Note that, hereinafter, the terms of driver and demand as well as the terms of parking space and supply are used interchangeably. We use pronouns "she" or "her" and "he" or "his" to differ a driver and an owner.

Let us consider a specific match $m(i,j)$ and the corresponding parking scheme $(s_{i,j}, s'_{i,j})$. The driver i's traveling and parking actions by using parking space j are illustrated in Figure 3. Firstly, driver i leaves from o_i and parks her car at r_j from time $s_{i,j}$. Secondly, she walks to d_i and stays there for a duration e_i. After that she walks back to r_j before time $s'_{i,j}$, and drives her car back to o_i. The driving time between o_i and r_j is $t_{i,j}$, the walking time between r_j and d_i is $t'_{i,j}$, and the directly driving time between o_i and d_i is t_i. Note that parameters $t_{i,j}$, $t'_{i,j}$, and t_i are given constants estimated with advanced real-time transportation information and navigation systems, such as Google maps.

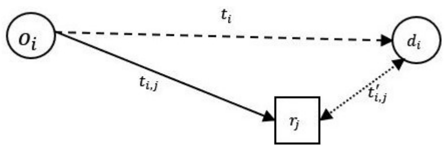


Figure 3: The spatial and temporal relation in match $m(i,j)$

For a match $m(i,j)$, the corresponding travel cost of driver i is the sum of her round trip cost from the origin to the destination via. space j and the corresponding parking fee, which is defined as

$$c_{i,j} = 2\alpha t_{i,j} + 2\beta t'_{i,j} + \gamma w_{i,j}, \tag{1}$$

where parameters α and β are travel cost per unit time for driving and walking, respectively, parameter γ is the parking fee per unit time, and $w_{i,j}$ denotes the parking duration of driver i at space j. It is clear that a driver's travel cost $c_{i,j}$ is determined by the location of the matched space and the corresponding parking duration.

We assume that on-time arrival is the main consideration of the drivers in our study. Thus, they will consider either self-driving with a confirmed parking space or taking taxi. In other words, on-street parking which requires uncertain time for searching and waiting is not an alternative to be considered in our model. The corresponding taxi fee of her round trip from the origin to the destination is formulated as follows.

$$c_i^0 = 2(\psi + \theta \max(0, t_i - t^0)), \tag{2}$$

where ψ is the fixed minimum fee, also known as the flag-down fare in the taxi industry, θ is the price per unit time in taxi, and $\max(0, t_i - t^0)$ is the traveling time exceeding the flag-down time t^0. Equation (2) implies that a driver just pays a fixed minimum fee ψ if the traveling time is shorter than the flag-down time, namely $t_i < t^0$. Besides, the waiting time costs of a taxi traveling to pick up the driver at the origin and the destination are neglected because this study assumes that a driver can reserve a taxi before the travel using an e-hailing App or a dial-hailing method.

As mentioned in the Introduction, the parking-sharing program is a typical bilateral platform and needs to attract as many drivers and owners as possible in order to reach the economies of scale. Drivers in most applications focus on their travel costs via shared parking spaces. Hence, the problem under investigation is to determine an appropriate matching-and-scheduling solution to maximize the sum of the travel cost savings of the drivers in the current list, compared with the taxi fee from the perspective of operational management. Note that similar objective functions have been considered in the literature [23, 30, 31]. Also, we realize that there exist other important indicators, such as the utilization rate of shared parking spaces, and the ratio of successfully matched drivers. We discuss these indicators in the simulation experiments of this paper.

2.3. Problem analysis

First of all, the parking time of driver i at parking space j is $w_{i,j} \equiv 2t^{'}_{i,j} + e_i$, and the ending time $s'_{i,j}$ is calculated by $s'_{i,j} = s_{i,j} + w_{i,j}$. Hence, in the following formulation, we only need to determine the matching $m(i,j)$ and the corresponding parking starting time $s_{i,j}$.

Secondly, to respect the scales of feasible matching conditions, the following Constraints (3)-(4) are used to filter infeasible parking spaces from original set J for each driver $i, i \in I$. We say that the following two necessary conditions must hold if space j is feasible for driver i.

$$\max(a_i + t_{i,j}, h_j) + t^{'}_{i,j} \le b_i, \tag{3}$$

$$\max(a_i + t_{i,j}, h_j) + w_{i,j} \leq k_j. \tag{4}$$

Constraint (3) implies that driver i's earliest arrival time at her destination (i.e., $\max(a_i + t_{i,j}, h_j) + t^{'}_{i,j}$) cannot exceed the required latest arrival time (i.e., b_i). Constraints (4) means that driver i's earliest departure time from space j (i.e., $\max(a_i + t_{i,j}, h_j) + w_{i,j}$) cannot exceed the upper bound of the available time interval (i.e, k_j). Consequently, for each driver i, the set of feasible spaces is constructed as J_i, where $J_i \subseteq J$ and for each parking space j, the set of of feasible drivers is constructed as I_j, where $I_j \subseteq I$. The following problem formulation is established based on the identified feasible sets I_j and J_i.

Before the formal formulation, the main notations are listed in Table 1.

3. Mathematical formulation

This section proposes an MIP model for the stated problem in Section 2 and analyzes the computational complexity of the problem.

3.1. Integrated matching-and-scheduling model

To formulate the problem, two classes of constraints are considered. The first one is the class of *matching* constraints, which ensure that the parking time of a driver does not exceed the available time interval of the matched space. The second one is the class of *scheduling* constraints, which ensure that there are no conflicts for any two drivers' parking times if both are assigned to the same parking space. The integrated optimal matching-and-scheduling problem is formulated as follows.

$$\textbf{\textit{Problem}}\ P \quad \max CS = \max \sum_{i \in I} \sum_{j \in J_i} (c_i^0 - c_{i,j}) x_{i,j}, \tag{5}$$

$$s.t. \quad \sum_{j \in J_i} x_{i,j} \leq 1,\ \forall i \in I, \tag{6}$$

$$s_{i,j} \geq \max(a_i + t_{i,j}, h_j) x_{i,j},\ \forall j \in J_i, \forall i \in I, \tag{7}$$

$$s_{i,j} \leq \min(b_i - t'_{i,j}, k_j - w_{i,j}) x_{i,j}, \forall j \in J_i, \forall i \in I, \tag{8}$$

Table 1: Main notations

Parameters	
I:	set of demands (i.e., drivers) for period T;
J:	set of supplies (i.e.,parking spaces) for period T;
I_j	set of feasible drivers for parking spaces j, $j \in J$;
J_i	set of feasible parking spaces for driver i, $i \in I$;
o_i, d_i:	origin and destination of driver $i, i \in I$, respectively;
a_i:	earliest departure time from origin o_i, $i \in I$;
b_i:	latest arrival time at destination d_i, $i \in I$;
e_i:	estimated staying duration at destination $d_i, i \in I$;
r_j:	location of parking space $j, j \in J$;
$[h_j, k_j]$:	available time interval of parking space $j, j \in J$;
$t_{i,j}$:	driving time between o_i and r_j, $i \in I$ and $j \in J$;
$t'_{i,j}$:	walking time between r_j and d_i, $i \in I$;
t_i:	driving time between o_i and d_i, $i \in I$;
$w_{i,j}$:	parking time of driver i at parking space j, $i \in I$ and $j \in J$;
α:	travel cost per unit time for self-driving;
β:	travel cost per unit time for walking;
γ:	parking fee per unit time;
ψ:	minimum constant taxi fee, also known as the flag-down fare;
θ	taxi price per unit time;
c_i^0:	travel cost of driver i by taking a taxi, $i \in I$;
$c_{i,j}$:	travel cost of driver i by self-driving and using parking space j, $i \in I$ and $j \in J$;
G	large enough value;
Decision variables	
$x_{i,j}$	binary integer variable, $x_{i,j} = 1$, if driver i is matched with parking space j; $x_{i,j} = 0$, otherwise, for $i \in I_j, j \in J$;
$y_{i,i'}$	binary integer variable, $y_{i,i'} = 1$, if driver i parks before driver i' at the same parking space; $y_{i,i'} = 0$, otherwise, for $i < i', i, i' \in I$.

$$s_{i',j} \geq s_{i,j} + w_{i,j} - (3 - x_{i,j} - x_{i',j} - y_{i,i'})G, \forall i, i' \in I_j, \forall j \in J, \quad (9)$$

$$s_{i,j} \geq s_{i',j} + w_{i',j} - (2 - x_{i,j} - x_{i',j} + y_{i,i'})G, i < i', i, i' \in I_j, \forall j \in J, \quad (10)$$

$$s_{i,j}, s_{i',j} \geq 0, x_{i,j} \in \{0,1\}, \text{and } y_{i,i'} \in \{0,1\}, i < i', i, i' \in I_j, \forall j \in J. \quad (11)$$

In the model above, the objective function (5) is to maximize the total travel cost saving (i.e, CS) of all drivers. Note that driver i may have a negative travel cost saving by using a shared parking slot if the drivers travel cost via a shared parking slot exceeds the taxi fee, i.e., $c_i^0 < c_{i,j}$. This case usually happens when the drivers travel time/distance is so short that the taxi cost in Equation (2) is reduced to the minimum fixed cost 2ψ and meanwhile the driver's parking fee is relatively high. In such a case, the mixed integer programming approach will not assign the parking space to the driver, since such assignment will worsen the objective value. Constraints (6) guarantee that any driver is assigned to at most one space. The *matching* Constraints (7)-(8) require that the starting time of driver i parking at space j cannot be earlier than the possible earliest starting time (i.e., $\max(a_i + t_{i,j}, h_j)$) or later than the possible latest ending time (i.e., $\min(b_i - t'_{i,j}, k_j - w_{i,j})$) if driver i is matched with space j (i.e, $x_{i,j} = 1$). More specifically, in the right term of Constraint (8), $b_i - t'_{i,j}$ is the latest starting time at space j with respect to the requested arrival time at the destination and $k_j - w_{i,j}$ is the latest starting times with respect to the upper bound of the available time interval. The *scheduling* Constraints (9)-(10) jointly guarantee that the parking time of driver i does not conflict with that of another driver i', if the two drivers are assigned to space j (i.e., $x_{i',j} = x_{i,j} = 1$). More specifically, Constraint (9) ensures that the starting time of driver i' using space j cannot be earlier than the ending time of driver i, if driver i uses the space before driver i'. Constraint (10) ensures that the starting time of driver i using space j cannot be earlier than the ending time of driver i', if driver i' uses the space before driver i. Note that if the two drivers are not assigned to space j, namely, $x_{i',j} + x_{i,j} < 2$, the two constraints always hold because the big G makes the right terms to be small enough values regardless whether $y_{i,i'} = 1$ or 0. Constraints (11) are restrictions on the decision variables. When an optimal solution to the MIP model P is obtained, the corresponding optimal matching solution and scheduling scheme are constructed as $M^* = \{m(i,j)|x_{i,j} = 1, \forall i \in I_j, j \in J\}$ and $S^* = \{(s_{i,j}, s'_{i,j} = s_{i,j} + w_{i,j})|\forall m(i,j) \in M^*\}$, respectively.

3.2. Complexity analysis

The computational time complexity of the problem is presented in following Theorem 1.

Theorem 1. *Problem P formulated by Constraints (5)-(11) is strongly NP-hard.*

Proof: We first construct a decision version of Problem P, defined as "DVP: Is there a feasible solution to Problem P ?". We then reduce the decision version of a classical parallel machines scheduling problem [22, 17], known as a strongly *NP-complete* one [8], into an instances of the constructed DVP. Thus Problem P is strongly *NP-hard*.

Firstly, let us introduce the decision version of a classical parallel machines scheduling problem, denoted as $P||C_{max}$. There is $|I|$ different jobs to be processed on $|J|$ identical parallel machines. For each job $i, 1 \leq i \leq |I|$, its processing time p_i is independent of the machines. The decision version of $P||C_{max}$ is that whether we can assign all jobs to parallel machines so that the maximum completion time of each machine does not exceed a given parameter C^U.

Now, consider a special instance of problem P with $|J|$ parking spaces each of which has an available time interval $[0, C^U]$ and $|I|$ drivers each of which has a parking duration $w_{i,j} = p_i$ on all spaces. Then the decision version of the special instance is whether there exists a feasible solution that assigns all drivers to spaces within time interval $[0, C^U]$. If we consider parking spaces as parallel machines and drivers' demands as jobs, the decision version of the above special instance of Problem P is equivalent to the decision version of $P||C_{max}$.

As it has been known that the decision version of $P||C_{max}$ is strongly *NP-complete* [8], the decision version of Problem P is also strongly *NP-complete.*

Consequently, Problem P is strongly *NP-hard*. $\square$

4. Iterative two-stage heuristic algorithm

Because of the strong NP-hardness nature of the problem P, it may result in unacceptable computational time if one applies an MIP solver, for example, ILOG-CPLEX, to solve middle- and large-sized instances of Problem P. In this section, an iterative two-stage heuristic algorithm (two-stage algorithm for short) is developed to fast find a near-optimal solution to Problem P under a real-time computational environment.

In the first stage of the algorithm, a matching problem, denoted as P^R, is formulated by relaxing partial matching and scheduling constraints from model P. The matching problem P^R is equivalent to a minimum-cost flow problem in a directed network and can be solved by a revised capacitated minimum-cost flow algorithm [18]. In the second stage, with the generated matching solution M^R, a straightforward but efficient heuristic method is proposed to schedule the matched drivers' demands of a same space. Note that the drivers' demands can not be successfully scheduled if the conflicts of these drivers' parking times cannot be avoided. If that happens, the algorithm adjusts the initial matching solution based on the identified conflicts and tries to schedule the drivers' demands once more. The matching and scheduling process repeats until all matched drivers being successfully scheduled or no parking spaces being available for the unscheduled drivers. The framework of our two-stage algorithm is presented in Figure 4. In the following, we provide the details of the two stages.

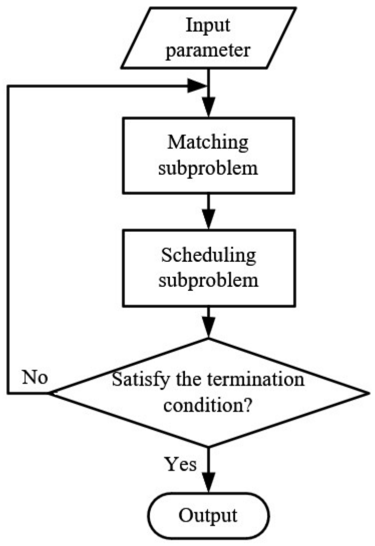


Figure 4: Framework of the two-stage algorithm

4.1. Matching stage

In this stage, the original model P is relaxed by removing Constraints (7)-(11) and adding additional capacity Constraint (12).

$$\textbf{\textit{Problem } } \boldsymbol{P^R} \quad \max \sum_{i\in I} \sum_{j \in J_i} (c_i^0 - c_{i,j}) x_{i,j}$$

$$s.t. \quad \text{Constraint (6)}$$

$$\sum_{i \in I_j} w_{i,j} x_{i,j} \le (k_j - h_j) \equiv W_j. \tag{12}$$

$$x_{i,j} \in \{0,1\},\ i \in I_j, j \in J. \tag{13}$$

Constraint (12) requires that the sum of drivers' parking durations not exceed the length of the available time interval of parking space j (i.e.,W_j) if these drivers are assigned to parking space j.

To efficiently solve the matching problem, model P^R is reduced to a network flow problem [3, 9, 19]. Let $G(V, A)$ be a directed network as depicted in Figure 5, where $V = I \cup J \cup \{s\} \cup \{t\}$ denotes the set of vertices consisting of drivers

I, parking spaces J, a source vertex s and a sink vertex t, and $A = \{(i,j)|i \in I,\ j \in J\} \cup \{(s,i)|i \in I\} \cup \{(j,t)|j \in J\}$ denotes the set of arcs. An arc $(i,j) \in A$, associated with a flow capacity $w_{i,j}$, connects vertex i to vertex j. If there is a flow with volume $l_{i,j}$, $0 \leq l_{i,j} \leq w_{i,j}$, through arc (i,j), it generates a cost $\pi_{i,j} = \frac{c_{i,j} - c_i^0}{w_{i,j}} l_{i,j} < 0$. Especially, we set that an arc(s,i) has a capacity $w_{s,i} = \max_{j \in J_i} w_{i,j}$ and generates zero cost with an arbitrary volume of flow, and an arc(j,t) has capacity $w_{j,t} = k_j - h_j \equiv W_j$ and generates zero cost with an arbitrary volume of flow.

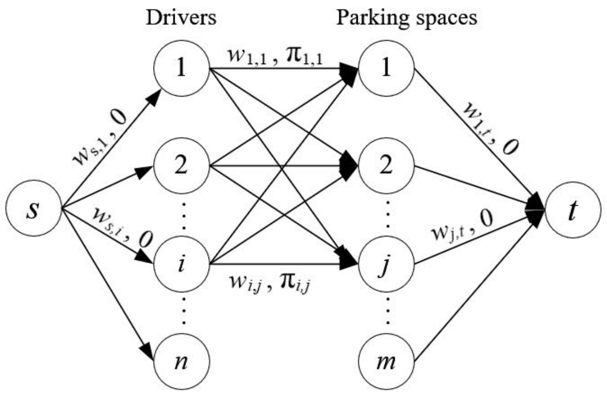


Figure 5: Minimum-cost flow in network $G(V, A)$

The following specific requirements are added to the network graph, with respect to the constraints of problem P^R. First, considering Constraints (6), in the feasible network flow, there is at most one outgoing flow from a driver vertex i to all space vertexes j, $\forall j \in J_i$; while a space vertex j can have more than one entering flow from driver vertexes $i, i \in I_j$. Second, with respect to Constraints (12), the total amount of flow through an arc$(j,t), \forall j \in J$, cannot exceed its capacity W_j, namely $\sum_{i \in I_j} l_{i,t} \leq W_j, \forall j \in J$. Third, with the binary variable Constraints (13), the amount of flow through arc(i,j), $i \in I_j, \forall j \in J$, is set as $l_{i,j} = w_{i,j}$ if $x_{i,j} = 1$ or $l_{i,j} = 0$ otherwise. Similarly, the amount of flow through arc(s,i), $\forall i \in I$, is set as $l_{s,i} \in \{0, w_{s,i} | \forall j \in J_i\}$ and the amount of flow through arc$(j,t), \forall j \in J$, is set as $l_{j,t} \in \{\sum_{i \in I'_j} w_{i,j} | \forall I'_j \subseteq I_j\}$. The flow balance constraints on any a vertex is required, which implies that the entering flow of a vertex equals its outgoing flow, except vertexes s and t. Based on the network established, a matching solution M^R is equivalent to flows from vertex s to vertex t through network $G(V, A)$, where an arc (i, j) with flow $l_{i,j} = w_{i,j}, i \in I_j, \forall j \in J$, corresponds to a match $m(i, j)$ in M^R. Therefore, the matching problem P^R is reduced to the capacitied minimum-cost flow problem in network $G(V, A)$ considering the above specific requirements.

Our proposed algorithm derived from the capacitied minimum-cost flow algorithm [18] successively identifies augmenting paths f^* with the current maximum cost from vertex s to vertex t. For an introduction on the flow augmenting path in a network flow problem, please refer to [29, 2]. Here, we briefly introduce the framework of the algorithm. First, a residual network $G_{f^*}(V, A)$ is initialized as the same one as the network $G(V, A)$. Then the minimum-cost augmenting path is found in $G_{f^*}(V, A)$ as follows. For the arcs in the augmenting path, their flows in the network $G(V, A)$ are augmented. Also, the residual network $G_{f^*}(V, A)$ is updated by calculating the residual capacity of arcs, moving arcs with zero residual capacities and adding reserve arcs. The processing continues until there is no new augmenting path in $G_{f^*}(V, A)$. Compared with the existing minimum-cost flow algorithm [18], our algorithm has the following two revisions.

Firstly, we give the following definition of an augmenting path in residual network $G_{f^*}(V, A)$.

Definition (augmenting path). A path f from vertex s to vertex t is referred to as an augmenting path if the cost of the path satisfies $\sum_{(i,j)\in f} \pi_{i,j} < 0$ and the capacity of the ending arc(j, t) satisfies $w_{j,t} > \sum_{(i,j)\in u^+} w_{i,j} - \sum_{(i,j)\in u^-} w_{j,i}$, where u^+ (resp. u^-) is the set of forwards arcs (resp. reverse arcs) from driver vertexes $i \in I_j$ (resp. parking space vertex $j \in J$) to the parking space vertex j (resp. driver vertexes i).

Hence, a minimum-cost augmenting path, denoted as f^*, is identified as

$$f^* = \arg\min_{f\in F} \sum_{(i,j)\in f} \pi_{i,j}, \tag{14}$$

where F is the set of augmenting paths in $G_{f^*}(V, A)$.

Secondly, the flow augmentation on the identified augmenting path f^* is calculated as follows. In the network $G(V, A)$, for an arc $(i, j) \in f^*, j \neq t$ and the ending arc $(j, t) \in f^*$, the flow are augmented as $l_{i,j} = w_{i,j}$ and $l_{j,t} = l_{j,t} + \sum_{(i,j)\in u^+} w_{i,j} - \sum_{(i,j)\in u^-} w_{j,i}$, respectively.

For the detailed steps of the capacities minimum-cost flow algorithm, please refer to [18].

4.2. Scheduling stage

This subsection schedules the parking times of the matched drivers in M^R according to the following heuristic rule. Considering objective function (5), we introduce the following metric to determine the priorities of the matched drivers of using the same parking space, which reflects a driver's contribution ratio to the objective value of Problem P.

$$\epsilon_i(j) = \frac{c_i^0 - c_{i,j}}{w_{i,j}},\ m(i, j) \in M^R. \tag{15}$$

where $\epsilon_i(j)$ is the unit cost-saving factor of driver i at parking space j. Driver i with a larger value of $\epsilon_i(j)$ contributes more value to the objective value per parking-time unit. Thus she has a higher priority of using the parking space compared to other competing drivers. All matched drivers in M^R to space j are sorted in an ordered list $\Psi_j = \{[1], [2], ..|\Psi_j|\}$ with non-increasing $\epsilon_i(j)$, where $\epsilon_{[1]}(j) \geq \epsilon_{[2]}(j) \geq ... \geq \epsilon_{[|\Psi_j|]}(j)$.

For each space j, we sequentially insert the parking demands of the matched drivers into its available time interval according to the order in Ψ_j. Accordingly, the starting and ending times of driver [1]'s parking are determined as $s_{[1],j} = \max(a_{[1]} + t_{[1],j}, h_j)$ and $s'_{[1],j} = s_{[1],j} + w_{[1],j}$. The starting and ending times of others are calculated as

$$s_{[i],j} = \max\left(a_{[i]} + t_{[i],j}, s'_{[i-1],j}\right), \text{and}$$
$$s'_{[i],j} = s_{[i],j} + w_{[i],j},\ 2 \leq i \leq |\Psi_j|. \tag{16}$$

We say that driver $[i]$ is not successfully scheduled at space j if $s'_{[i],j} \geq k_j$, $[i] \in \Psi_j$, which implies that there is no enough available time for driver $[i]$'s parking demand. Subsequently, the unsuccessfully scheduled driver is added to the set of remaining drivers, denoted as Φ, and will be re-matched in the next iteration. Note that space j may be shared by multiple drivers, whose scheduled parking duration $(s_{[i],j}, s'_{[i],j})$ divide the available interval $[h_j, k_j]$ into several subintervals, denoted as $[h_j^l, k_j^l]$. In the next iteration, we can update parking space set J with some dummy parking spaces, each of which has an available time interval as $[h_j^l, k_j^l]$ and the same locations. The detailed steps of scheduling procedure is presented in lines 5-22 of Algorithm 1 in the following subsection.

4.3. Steps of the two-stage algorithm

Integrating the matching and scheduling procedures in the above two subsections, the steps of the two-stage algorithm are presented in Algorithm 1 below. The algorithm iteratively improves the matching-and-scheduling solution to approximate the optimal one, though it solves the matching and scheduling subproblems independently. In particular, the algorithm firstly generates a trial matching solution M^R by relaxing the scheduling constraints (line 2 of Algorithm 1). For each iteration (lines 3 to 28 of Algorithm 1), the algorithm sequentially schedules the parking times of the matched drivers in M^R at the same space j according to order Ψ_j in subsection 4.2 (lines 6-16 of Algorithm 1). The unsuccessfully scheduled drivers are added to set Φ because there is no enough available time to meet their parking demands (line 18 of Algorithm 1). These unsuccessfully scheduled drivers will be matched and scheduled again at other parking spaces in the next iteration. The available interval of the parking space will be updated according to the scheduled parking durations of the successfully scheduled drivers (line 20 of Algorithm 1). The algorithm iteratively run matching and scheduling procedures until all matched drivers are successfully scheduled (line 23 of Algorithm 1) or no parking space is available to meet the unscheduled drivers' demands (line 3 of Algorithm 1). The numerical experimental results in Section 5.2 indicate that the final matching-and-scheduling solutions approximate closely the optimal solutions of the instances tested.

Algorithm 1 : Steps of the two-stage algorithm

```
1:  **Initialize:** matching solution $M \leftarrow \varnothing$, and scheduling solution $S \leftarrow \varnothing$;
2:  Generate a trial matching solution $M^R$ to model $P^R$ by the revised capacitied minimum-cost flow algorithm;
3:  **while** $M^R \neq \varnothing$ **do**
4:      Set $\Phi \leftarrow \varnothing$;
5:      **for** each parking space $j$ in $M^R$ **do**
6:          Sort the drivers matched to parking space $j$ in the non-descending order $\Psi_j = \{[1], [2], ..[|\Psi_j|]\}$ in terms of metric $\epsilon_i(j)$ in Equation (15);
7:          Set $s_{[1],j} \leftarrow \max(a_{[1]} + t_{[1],j}, h_j)$ and $s'_{[1],j} \leftarrow s_{[1],j} + w_{[1],j}$, for $[1] \in \Psi_j$;
8:          **if** $s'_{[1],j} \leq k_j$ **then**
9:              Add $m([1], j)$ to $M$ and $(s_{[1],j}, s'_{[1],j})$ to $S$;
10:         **else**
11:             Add driver $[1]$ to $\Phi$;
12:         **end if**
13:         **for** each driver $[i]$ in $\Psi_j$, $i = 2, 3, ..., |\Psi_j|$ **do**
14:             Set $s_{[i],j} \leftarrow \max(a_{[i]} + t_{[i],j}, s'_{[i-1],j})$ and $s'_{[i],j} \leftarrow s_{[i],j} + w_{[i],j}$;
15:             **if** $s'_{[i],j} \leq k_j$ **then**
16:                 Add $m([i], j)$ to $M$ and $(s_{[i],j}, s'_{[i],j})$ to $S$;
17:             **else**
18:                 Add driver $[i]$ to $\Phi$;
19:             **end if**
20:             Divide interval $[h_j, k_j]$ into several subintervals $[h^l_j, k^l_j]$ according to the scheduled parking duration $(s_{[i],j}, s'_{[i],j})$ in $S$;
21:         **end for**
22:     **end for**
23:     **if** $\Phi = \varnothing$ **then**
24:         Goto line 29;
25:     **end if**
26:     Set $I \leftarrow \Phi$ and update $J$ with subintervals $[h^l_j, k^l_j]$;
27:     Generate a new $M^R$ to model $P^R$ with the updated $I$ and $J$;
28: **end while**
29: Output $M$ and $S$.
```

The computational time of the two-stage algorithm depends mainly on the iterations consisting of lines 3-28 of Algorithm 1. Now let us consider the scheduling procedure in lines 5-22 of Algorithm 1 with the generated matching solution M^R. Line 6 of Algorithm 1 consumes $O(|\Psi_j| \log |\Psi_j|)$ time to construct sequence Ψ_j by sorting matched drivers for each parking space j. As there are at most $|I|$ drivers and $|J|$ parking spaces, the computational time of sorting matched drivers for all parking spaces is $O(\sum_{j=1}^{|J|}(|\Psi_j| \log |\Psi_j|)) \leq O(|I| \log |I|)$. Lines 7-22 of Algorithm 1 need at most $O(|I|)$ time to determine the starting times of all matched driver for all parking spaces and divide the available interval into at most $O(|I|)$ subintervals. Therefore, the computational time complexity of lines 5-22 of Algorithm 1 is $O(|I| \log |I|)$. Line 27 of Algorithm 1 calls the revised capacities minimum-cost flow algorithm to generate a trial matching solution. In the worst case, it needs at most B^U times to augment the flow from initial zero flow to the maximum flow in network $G(V, A)$, where $B^U \equiv \min(\sum_{i \in I_j, j \in J} w_{i,j}, \sum_{j \in J} W_j)$ is the upper bound of the flow in the network. The computational time for finding each augmenting path in network $G(V, A)$ is $O(|V||A|) \leq O(|I||J| + |J|^2)$. Therefore, line 27 needs at most $O(B^U(|I||J| + |J|^2))$ computational time. Based on the above analysis, the computational time of each iteration is at most $O(B^U(|I||J| + |J|^2) + |I|(\log |I|)$. The algorithm runs $|I|$ iterations in the worst case that only one driver is matched and scheduled in each iteration. Consequently, the total computational time complexity of the algorithm is at most $O(B^U|J|(|I|^2 + |I||J|) + |I|^2 \log |I|)$.

5. Simulation experiments

To examine the performance of the two-stage algorithm and the multi-to-one matching pattern, a simulation experiment bed is established partially based on the statistical results of the real data collected from “DingdingParking” application in a CBD of Beijing city. We firstly compare the two-stage algorithm with the state-of-the-art commercial MIP solver, CPLEX tool (version 12.80), in solving single-period instances. For the performance evaluation of the proposed parking reservation system with the multiple-to-one matching pattern, a multi-period scenario for five weekdays is generated. Three metrics: The total travel cost saving of drivers (i.e., the optimization objective of the problem), the fulfillment rate of demands, and the utilization rate of shared parking spaces, are recorded and compared with those provided by the existing one-to-one matching pattern.

5.1. The simulation experiment bed

We collect the real data set consisting of 1609 demand records and 995 supply records from "DingdingParking" application. We find that the announcement times of drivers are concentrated in three phases: Early morning (about from 7:00 am to 9:00 am), late morning (about from 9:00 am to 12:00 am), and afternoon (about from 2:00 pm to 6:00 pm). Also the announcement times of owners are concentrated in three phases: Early morning (about from 6:00 am to 10:00 am), late morning (about from 10:00 am to 12:00 am), and afternoon (about from 1:00 pm to 3:00 pm). Accordingly, we classify drivers and owners into three types, respectively indexed as driver types I, II, and III, and owner types I, II and II. Type I drivers and type I owners are mostly commuters with regular travel needs, who submit their demands or supplies in early morning and have relatively long parking times or available time intervals. Types II and III drivers and types II and III owners are identified as irregular travelers with occasional travel purposes, who send their demands or supplies in late morning and afternoon, and have relatively short parking time or available time interval. The detailed statistical description of the three types of drivers and owners are presented in Appendix A.1.

A simulation experiment bed is established to simulate the demands and supplies submitted by drivers and owners respectively through the horizon from 6:00 am to 6:00 pm for five weekdays. The announcement times of the three types demands and supplies are respectively generated by two independent *Poisson* processes with the different announcement rates listed in Tables A.3 and A.4. Note that the temporal parameters of demands: The earliest departure time

(a_i), the latest arrival time (b_i), and the estimated staying duration (e_i) are not recorded by the current "DingdingParking" application. We cannot access to the spatial parameters o_i, d_i, and r_j either, due to the users' privacy protection rule of the application. To deal with these gaps, our experiment assumes that these missing parameters are subjected to normal probability distributions and estimate them by the collected data. The detailed methods are introduced in Appendix A.2.

With the generated demand $D_i(o_i, d_i, a_i, b_i, e_i)$ and supply $E_j(r_j, h_j, k_j)$, for each driver i, her traveling time between o_i and r_j is calculated as $t_{i,j} = line(o_i, r_j)/v$, where $line(o_i, r_j)$ is the airline distance between o_i and r_j and $v = 0.60\ km/min$ is the average vehicle speed. Her directly traveling time between o_i and d_i is $t_i = line(o_i, d_i)/v$. The walking time between r_j to d_i is calculated as $t'_{i,j} = line(r_j, d_i)/v'$, where $v' = 0.083\ km/min$ is the average walking speed. Last, the travel cost parameters are set as $\alpha = 0.50\ yuan/min$, $\beta = 2\ yuan/min$, $\gamma = 0.05\ yuan/min$, $\theta = 1.20\ yuan/min$, $\psi = 10\ yuan$, and $t^0 = 5\ mins$ in this experiment.

5.2. Computational results of single-period instances

In this subsection, 25 groups of single-period instances are generated with various sizes: $|I|, |J| = 10,\ 20,\ 30,\ 40,$ and 50, by the simulation bed in subsection 5.1. The two-stage algorithm and CPLEX are applied to solve the instances generated. We let CPLEX output an initial feasible solution (denoted as (M^0, S^0)) and the optimal solution (denoted as (M^*, S^*)). Note that due to the NP-nature of problem, we terminate CPLEX and let it output the relaxed solution with the best upper bound so far if CPLEX cannot find an optimal solution within a computational time limit equaling one hour. The computational times of the two-stage algorithm and CPLEX, the values of *Gap* of the near-optimal solutions (denoted as (M^H, S^H)) generated by the two-stage heuristic algorithm from (M^*, S^*) as well as the values of *Gap* of the initial feasible solution (M^0, S^0) from (M^*, S^*) are listed in Table 2. Besides, we report the ratio of size of feasible matching pairs filtered by Constraints (3)-(4) to the size of

Table 2: Performance of the two-stage algorithm and CPLEX

Groups of instances ($\lvert I\rvert * \lvert J\rvert$)	Size of problem reduced RS(%)	(M^*, S^*) by CPLEX		(M^0, S^0) by CPLEX		(M^H, S^H) by two-stage algorithm	
		CPU time (seconds)	Number of solvable instance	CPU time (seconds)	Gaps (%)	CPU time (seconds)	Gaps (%)
10*10	86.00	0.05	50	0.04	55.38	0.04	2.50
10*20	85.06	0.16	50	0.04	60.90	0.04	6.39
10*30	85.53	0.25	50	0.04	63.73	0.06	7.01
10*40	83.62	0.15	50	0.04	72.67	0.04	4.01
10*50	85.35	0.19	50	0.04	73.53	0.06	3.22
20*10	84.91	0.26	50	0.04	64.65	0.03	6.17
20*20	88.75	0.62	50	0.04	71.99	0.09	4.90
20*30	87.64	0.56	50	0.04	71.84	0.06	8.54
20*40	86.65	0.66	50	0.05	76.78	0.05	6.88
20*50	86.96	0.31	50	0.05	77.98	0.05	5.44
30*10	86.10	0.65	50	0.04	72.32	0.03	7.21
30*20	87.47	2.49	50	0.04	76.57	0.04	8.79
30*30	89.28	5.55	50	0.05	78.48	0.06	8.73
30*40	86.24	6.32	50	0.06	81.66	0.06	8.53
30*50	87.37	9.97	50	0.06	81.80	0.08	8.28
40*10	84.64	8.62	50	0.04	76.36	0.04	5.78
40*20	86.14	41.07	50	0.05	80.01	0.07	10.95
40*30	87.05	92.53	50	0.06	82.25	0.07	10.56
40*40	91.31	482.32	45	0.07	83.68	0.14	9.52
40*50	87.17	1147.69	36	0.08	84.78	0.10	8.75
50*10	84.30	59.39	50	0.05	80.31	0.05	5.70
50*20	86.62	296.68	48	0.06	81.89	0.10	9.25
50*30	87.15	609.14	39	0.06	83.73	0.24	13.78
50*40	86.50	1466.45	31	0.07	83.74	0.39	14.28
50*50	87.28	1797.35	28	0.09	83.75	0.65	13.18
Average	**86.60**	**241.18**	**47**	**0.05**	**76.03**	**0.11**	**7.93**

demand and supply, denoted as RS, and the numbers of instances in which an optimal solution can be found within one hour by CPLEX in Table 2 as well. The calculation methods of Gap and RS are respectively

$$Gap = \frac{CS((M^*, S^*)) - CS(\cdot)}{CS((M^*, S^*))} \times 100\%, \text{and}$$

$$RS = \frac{\sum_{j \in J} |I_j| + \sum_{i \in I} |J_i|}{2|I||J|} \times 100\%,$$

where $CS((M^*, S^*))$ is the total travel cost saving by (M^*, S^*), and $CS(\cdot)$ is the total travel cost saving by either (M^H, S^H) or (M^0, S^0).

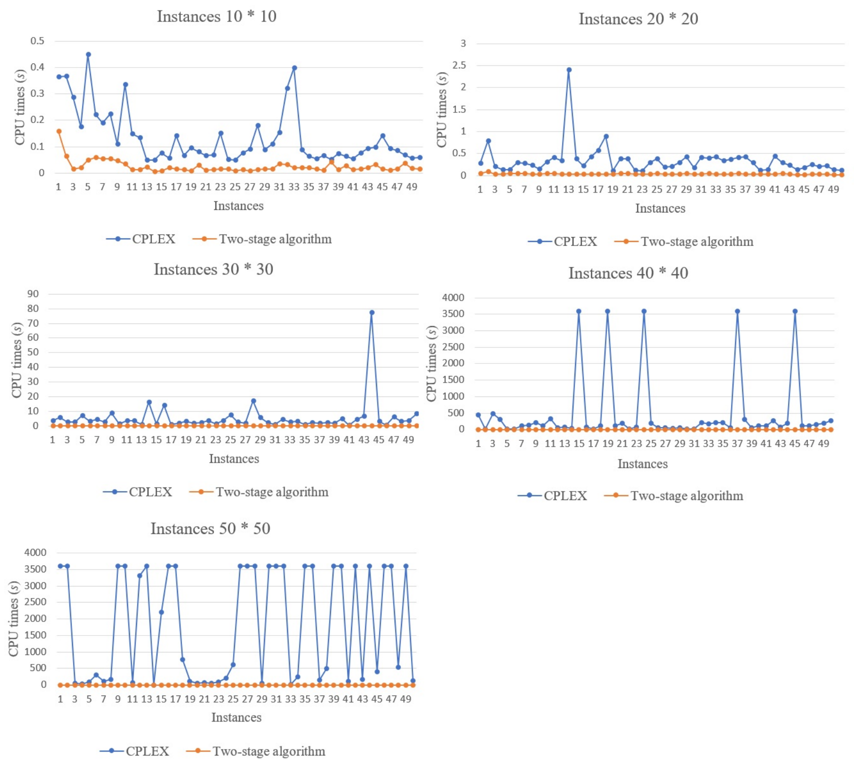


Figure 6: Computational times of five groups of instances with $|I| = |J|$

The quality of the solutions (i.e., "*Gap*") generated by the two-stage algorithm is shown in the 6^{th} column of Table 2. We can see that the two-stage algorithm can efficiently generate high-quality solutions for the instances of the 25 groups. The average gap is 7.93% for all instances, even though the best upper bounds so far are used for the calculation of gaps of some large-size instances. Furthermore, for the instances with small-size with $|I|$=10, 20, 30, and 40, and $|J|$=10, 20, and 30, CPLEX can generate optimal solutions and the gaps do not exceeded 11%. For some instances with large-size, CPLEX cannot yield optimal solutions within one hour due to the NP-hard nature of the problem, the gaps of larger-size instances become larger because of bad upper bounds provided by CPLEX. Besides, we analyze the effect of the volumes of parking demand and supply on the performance of the two-stage algorithm. The algorithm generates relatively better solutions with smaller value of gaps to the instances when $|I|$ is much larger or smaller than $|J|$, while the algorithm has slightly larger values of gaps to the instances with $|I|$ closing to $|J|$, reflecting the general balance of parking demand and supply.

The computational times in Table 2 indicate that the two-stage algorithm runs much faster (less than one second in CPU time) than CPLEX for generating (M^*, S^*). On average, the computational time of the two-stage algorithm is only of 0.05 % of that consumed by CPLEX. The data in the column "*RS*" indicates that Constraints (3) and (4) reduce the sizes of matching pairs (i.e., I_j and J_i) and thus speeds up the algorithm. In addition, we present the CPU times consumed by the two-stage algorithm and CPLEX for 50 instances of five groups with instance sizes $|I| = |J| = 10,\ 20,\ 30,\ 40$, and 50 in Figure 6. The CPU times of the two-stage algorithm are relatively stable and small, compared with the CPU times of CPLEX. It implies that the computational efficiency of the two-stage algorithm is less affected by the parameters of instances. The main reason is that the algorithm uses a revised capacitied minimum-cost flow algorithm to generate a trial matching solution and a simple but efficient heuristic rule to schedule the matched demands. Comparing the computational times of the algorithm for solving instances with $|I| < |J|$ and $|I| > |J|$, the two-stage algorithm consumes relatively longer computational time to solve the instances with $\frac{|I|}{|J|} = \nu > 1$ than those with $\frac{|I|}{|J|} = \frac{1}{\nu}$. For example, the algorithm consumes 59.39 seconds for instances $50 * 10$, namely $\frac{|I|}{|J|} = 5$, while it consumes 0.19 seconds for the instances $10 * 50$, namely $\frac{|I|}{|J|} = 0.2$. In the instances with $\frac{|I|}{|J|} > 1$ implying that the parking demand exceeds the supply, the algorithm usually assigns more drivers to the same parking spaces. The algorithm thus needs more iterations and computational times to adjust the matching solution and then schedules the matched drivers' parking times.

In addition, we compare the performance of the two-stage algorithm with CPLEX for just generating initial feasible solutions to the problem instances. As shown in Table 2, comparing the corresponding terms "*Gap*" for each group, the solutions (M^H, S^H) generated by the two-stage algorithm are much better than the initial feasible solutions (M^0, S^0). The average gap of (M^0, S^0) for 25 groups is 76.03%, while that of (M^H, S^H) is 7.93%. On the other hand, the two-stage algorithm runs fast, with 0.11 seconds as the average computational time, which is only twice of the average computational time to generate (M^0, S^0) by CPLEX. To sum up, the results above indicate that the two-stage algorithm can efficiently generate near-optimal matching-and-scheduling solutions to the numerical instances with various sizes.

5.3. Computational results of a multi-period scenario

To simulate a multi-period scenario for five weekdays, the horizon from 6:00 am to 6:00 pm of each weekday is divided into $\mathbf{T} = 72$ periods with 10 minutes for each period. Instances of sizes 300*100, 300*150, and 300*200 are randomly generated with the simulation bed in subsection 5.1. As stated in Section 2, the reservation system makes use of the flexibility of drivers' trips to schedule the matched drivers at the same parking space. The flexibility of driver i's trip is defined as the slack time $\delta_i = b_i - a_i - t_i$. Thus, we use three slack times $\delta_i = 5,\ 15,$ and 25 minutes for all drivers' trips to investigate the effect of flexibility of drivers' trips on the performance of the system. In this experiment, we set the maximum slack time as 25 minutes because further increasing the number of slack time has little effect on the performance of the system.

The performances of our reservation system with one-to-one and multi-to-one matching patterns are evaluated by the following three metrics: Total travel cost savings of drivers (CS), the fulfillment rate of demands (SM), and the utilization rate of shared parking spaces (UR). The first one is the optimization objective of the problem, the second one measures the rate of drivers who successfully obtain parking spaces through the system and the third one measures the utilization of the shared parking spaces. The first two metrics reflect the benefits of drivers and the last one reflects the benefits of owners. The calculations of the three metrics are given as follows.

$$CS = \textstyle\sum_{T=1}^{72} \sum_{m(i,j)\in M_T} (c_i^0 - c_{i,j}),$$

where $\sum_{m(i,j)\in M_T}(c_i^0 - c_{i,j})$ is the total travel cost saving and M_T is the generated matching solution for each period T.

$$SM = \sum_{T=1}^{72} \frac{|M_T|}{|I_T|},$$

where $\frac{|M_T|}{|I_T|}$ is the ratio of demands satisfied (i.e., $|M_T|$) to the total demands(i.e., $|I_T|$) for each period T.

$$UR = \sum_{T=1}^{72} \sum_{i:m(i,j)\in M_T} w_{i,j} / \sum_{j\in J}(k_j - h_j),$$

where $\sum_{m(i,j)\in M_T} w_{i,j} / \sum_{j\in J}(k_j - h_j)$ is the utilization rate of the shared parking spaces for each period T.

The simulation results for the one-to-one and the multiple-to-one matching patterns are presented in Figures 7-9. Observing the values of CS in Figure 7, we find that the multiple-to-one matching pattern can save more travel cost for the drivers than the one-to-one matching pattern in all instances. Besides, the multiple-to-one matching pattern also outperforms the one-to-one matching pattern in terms of the other two metrics: SM and UR as illustrated in Figures 8 and 9. The average improvement rates are 17.25% and 8.08% in terms of SM and UR, respectively. The reason is that the multiple-to-one matching pattern can assign multiple drivers to the "best" parking space if the available time interval of the space can accommodate the parking demands of these drivers.

Furthermore, we analyze the three metrics CS, SM, and UR obtained by the multiple-to-one pattern with any given slack times δ_i. The values of CS and SM both increase but the values of UR decrease when private parking spaces are supplied by owners from $|J| = 100$ to 200. The intuition is that the benefits of drivers, measured by CS and SM, are improved if sufficient parking spaces are supplied by owners. However, the utilization rates of the shared parking spaces, measured by UR, are affected if excessive parking spaces are supplied even though the proposed algorithm can assign multiple drivers to use the same parking space. Therefore, from the perspective of utilization of parking spaces, it is a challenging task to balance the demands and supplies in a dynamic environment.

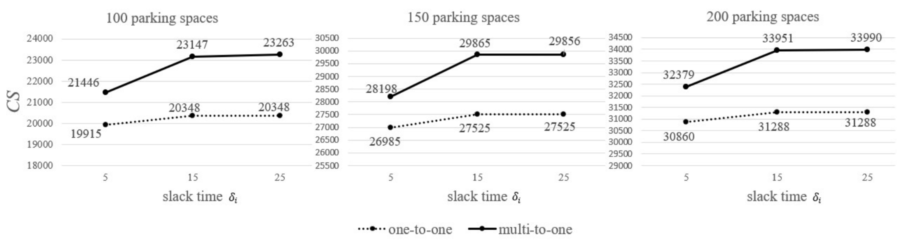


Figure 7: Travel cost savings (CS) with different slack times

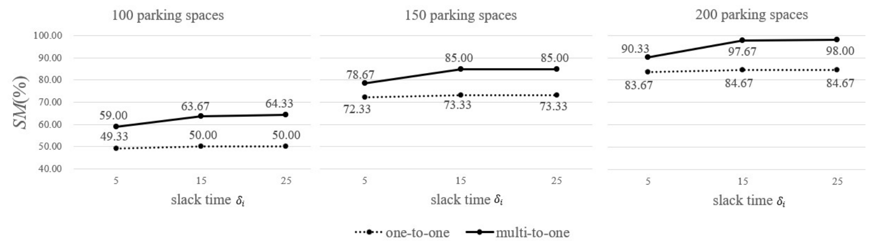


Figure 8: Fulfillment rates (SM) with different slack times

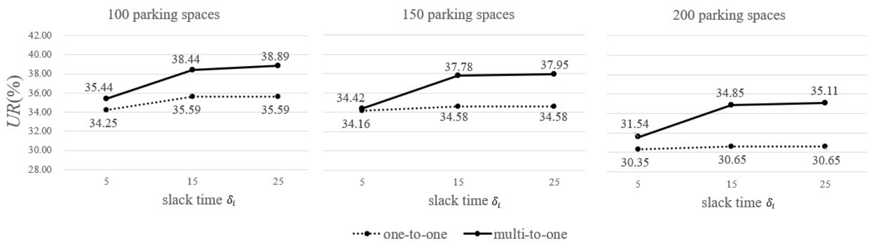


Figure 9: Utilization rates (UR) with different slack times

In addition, we investigate the effect of the flexibility of drivers' trips on the performance of the system. As presented in Figures 7 to 9 when the slack time δ_i increases from 5 to 25 minutes, all metrics obtained by both patterns are
665 improved, but the improvements of one-to-one matching pattern is relatively small. It indicates that the flexibility of drivers' trips has a positive effect on improving the performance of the parking reservation system. It is because that the system can use the slack times of driver trips to improve the integrated matching-and-scheduling procedure. The results also show that when the slack

time reaches a degree, $\delta_i = 15$ $mins$, further increasing the slack time to 25 $mins$ has little effect on improving the performance of the system. Thus, in parking-sharing management, to achieve a better performance of the system, we only need drivers to provide a certain degree of slack times of their trips.

To sum up, the computational results of single-period and multiple-period instances respectively demonstrate that our proposed two-stage algorithm outperforms the existing CPLEX tool and the multiple-to-one matching pattern dominates the one-to-one matching one.

6. Discussions

First of all, we note that limiting the supply of parking spaces, setting high parking price, and applying a tradable credit scheme are effective methods to reduce traffic congestion in many city centers [26, 27]. However, the aim of this study is to optimize the system performance at the operational level by saving the drivers' travel costs in parking-sharing services. Drivers taking part into a parking-sharing program usually have relatively strong self-driving demands. For example, drivers need to carry some heavy stuff with themselves to their destinations, they need to drop off some family members at schools or workplaces during their trips, or they have some physical problems so that they have to drive their cars to their destinations. The optimization objective of our problem may induce more travelers to select the self-driving alternative. In our further study, we plan to investigate the effect of parking-sharing programs on traffic congestion and develop an approach to trade off the travel cost of drivers and traffic congestion. In addition, this study considers the situation where on-time arrival is the main concern of the drivers, and therefore assumes that a driver without a reserved parking space will take taxi as the alternative, instead of looking for on-street parking slots. A future research topic is to develop an integrated travel demand management system in which multiple travel modes, such as on-street parking, parking-sharing, taxi, ridesharing, and public modes, are taken into account. In such a complicated problem, travelers' selection behaviors need to be investigated and modeled.

Secondly, note that the current systems, such as "DingdingParking" application, adopt the *first-book-first-serve* rule to confirm the drivers' parking reservations. It is obvious that such a system cannot achieve the optimum performance in most cases. This inspires us to develop a smarter matching-and-scheduling approach to improve the performance of the program. However, for the application of our proposed approach, the program needs additional information of driver travel plans, i.e., their earliest departure times, the latest arrival times and the estimated staying durations at the destinations. It is another interesting topic to encourage drivers to precisely and truthfully report such private information to the system. This topic is out of the scope of this study and currently addressed in another working paper.

Lastly, we note that some study on human mobility shows a high degree of temporal and spatial regularity from the view point of human pollution and the individual traveler's trajectories can be predicted by a probability distribution [11]. Our paper also obtains similar observations from the real data provided by "DingdingParking" application. Based on the results in Appendix A.1, drivers can be clustered into three types, as shown in Table A.3. The drivers of type I are mostly commuters who have high degrees of temporal and spatial regularity. They drive from home to workplaces with relatively fixed arrival times and parking durations for each weekday. However, the drivers of types II and III are mostly with occasional travel purposes, for shopping, entertainment, dealing with private affairs etc., during the late morning and afternoon. Similarly, owners can also be clustered into the three types, as shown in Table A.4. Therefore, our paper has proposed a real-time parking reservation approach to deal with such a complicated scenario. Our simulation experiments show that the proposed model and algorithm can deal with regular and irregular travel needs together. In practice, a parking reservation system can be easily adapted by pre-treating the regular drivers and owners with the early-bird-fee rule or a long-term contract firstly, then processing the uncertain demands and supplies of irregular drivers and owners.

7. Conclusions

To deal with the challenge of insufficient parking resources, many large cities have launched parking-sharing applications, which offer existing private parking spaces to nearby drivers with parking demands. This paper proposes a real-time parking reservation system for a parking-sharing program. To maximize the travel cost savings of drivers, an integrated matching-and-scheduling model is developed, which not only assigns drivers to feasible parking spaces but also schedules driver parking times. The optimization problem is proved to be strongly NP-hard and an iterative two-stage heuristics is developed to fast generate a near-optimal solution under a real-time setting. The experimental results of single- and multi-period instances demonstrate the efficiency and effectiveness of the proposed two-stage algorithm and multiple-to-one matching pattern comparing with the CPLEX tool and the existing one-to-one matching pattern, respectively.

Based on the experimental results in subsection 5.3, we have the following two operational insights. First, as a sharing economic model, a private parking-sharing application needs to encourage more and more drivers and owners to participate, in order to achieve the effect of a bilateral platform. However, the experimental results show that excessive supplies or demands can yield little improvement of the system performance. Furthermore, the un-matched owners or drivers may leave the platform and may not use it any more in future. As it is difficult to decide a perfect balance between the demands and supplies, since they join the system randomly, it may be reasonable to develop a dynamic pricing method to control the demands and supplies in a real-time manner. Second, the experimental results also show that a certain degree of slack times of drivers' trips has a meaningful effect on the performance of the system. It will be helpful if the operator can design an incentive mechanism to encourage drivers to provide their real slack times to the system. Obviously, the pricing method and the incentive mechanism mentioned above are out of the scope of this study, which will be addressed in future study.

Acknowledgements

The authors are very grateful to the five reviewers for their constructive comments and suggestions that have led to an improved version of this paper. Authors are also thankful for the data support of Tongyudao Technology Company ("DingdingParking" application). This research is partially funded by: Natural Science Foundation of China (Nos. 71971044, 71571032, 71531003, and 71432004), the Leading Talent Program of Guangdong Province (No. 2016LJ06D703), PHC XU GUANGQI Program 2019 (No. 43361PA), and Sichuan Province Science and Technology Program (No. 2020YJ0026).

References

[1] Bryan Blanc and Miguel Figliozzi. Modeling the impacts of facility type, trip characteristics, and trip stressors on cyclists comfort levels utilizing crowdsourced data. *Transportation Research Record*, 2587(1):100–108, 2016.

[2] John Adrian Bondy, Uppaluri Siva Ramachandra Murty, et al. *Graph theory with applications*, volume 290. Macmillan London, 1976.

[3] André Chassein and Anika Kinscherff. Complexity of strict robust integer minimum cost flow problems: An overview and further results. *Computers & Operations Research*, 104(4):228–238, 2019.

[4] Xiao Chen, Elizeu Santos-Neto, and Matei Ripeanu. Crowdsourcing for on-street smart parking. In *Proceedings of the Second ACM International Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, pages 1–8, New York, USA, 2012.

[5] Zhibin Chen, Zhengtian Xu, Mahmood Zangui, and Yafeng Yin. Analysis of advanced management of curbside parking. *Transportation Research Record*, 2567(1):57–66, 2016.

[6] Shuo-Yan Chou, Zaenal Arifin, and Anindhita Dewabharata. Optimal decision of reservation policy for private parking sharing system. In *Proceddings of 2018 International Conference on Computer, Control, Informatics and its Applications*, pages 66–71, Tangerang, Indonesia, 2018.

[7] Lili Du and Siyuan Gong. Stochastic Poisson game for an online decentralized and coordinated parking mechanism. *Transportation Research Part B: Methodological*, 87(5):44–63, 2016.

[8] Michael R. Garey and David S. Johnson. " Strong " NP-Completeness Results: Motivation, Examples, and Implications. *Journal of the ACM*, 25(3):499–508, 1978.

[9] Jean B. Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. A strongly polynomial contraction-expansion algorithm for network flow problems. *Computers & Operations Research*, 84:16–32, 2017.

[10] Yanfeng Geng and Christos G. Cassandras. New smart parking system based on resource allocation and reservations. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1129–1139, 2013.

[11] Marta C. González, Csar A. Hidalgo, and Albert-László Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[12] Wei Guo, Yi Zhang, Man Xu, et al. Parking Spaces Repurchase Strategy Design via Simulation Optimization. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 20(3):255–269, 2016.

[13] Eren Inci. A review of the economics of parking. *Economics of Transportation*, 4(1-2):50–63, 2015.

[14] Mor Kaspi, Tal Raviv, Michal Tzur, et al. Regulating vehicle sharing systems through parking reservation policies: Analysis and performance bounds. *European Journal of Operational Research*, 251(3):969 – 987, 2016.

[15] Wei Liu, Hai Yang, and Yafeng Yin. Expirable parking reservations for managing morning commute with parking space constraints. *Transportation Research Part C: Emerging Technologies*, 44(7):185–201, 2014.

[16] Wei Liu, Hai Yang, Yafeng Yin, et al. A novel permit scheme for managing parking competition and bottleneck congestion. *Transportation Research Part C: Emerging Technologies*, 44(7):265–281, 2014.

[17] Janis S. Neufeld, Jatinder N.D. Gupta, and Udo Buscher. A comprehensive review of flowshop group scheduling literature. *Computers & Operations Research*, 70(6):56–74, 2016.

[18] James B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[19] Leizer L. Pinto, Kátia C.C. Fernandes, Kleber V. Cardoso, et al. An exact and polynomial approach for a bi-objective integer programming problem regarding network flow routing. *Computers & Operations Research*, 106(6):28–35, 2019.

[20] Mireia Roca-Riu, Elena Fernndez, and Miquel Estrada. Parking slot assignment for urban distribution: Models and formulations. *Omega*, 57(12):157 – 175, 2015.

[21] Caroline J. Rodier and Susan A. Shaheen. Transit-based smart parking: An evaluation of the san francisco bay area field test. *Transportation Research Part C: Emerging Technologies*, 18(2):225–233, 2010.

[22] Hesam Shams and Nasser Salmasi. Parallel machine scheduling problem with preemptive jobs and transportation delay. *Computers & Operations Research*, 50(10):14–23, 2014.

[23] Chaoyi Shao, Hai Yang, Yi Zhang, et al. A simple reservation and allocation model of shared parking lots. *Transportation Research Part C: Emerging Technologies*, 71(10):303–312, 2016.

[24] Donald C. Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006.

[25] Dušan Teodorović and Panta Lučić. Intelligent parking systems. *European Journal of Operational Research*, 175(3):1666–1681, 2006.

[26] Guangmin Wang, Ziyou Gao, and Meng Xu. Integrating link-based discrete credit charging scheme into discrete network design problem. *European Journal of Operational Research*, 272(1):176–187, 2019.

[27] Guangmin Wang, Ziyou Gao, Meng Xu, et al. Models and a relaxation algorithm for continuous network design problem with a tradable credit scheme and equity constraints. *Computers & Operations Research*, 41(1):252–261, 2014.

[28] Xiaotian Wang and Xin Wang. Flexible parking reservation system and pricing: A continuum approximation approach. *Transportation Research Part B: Methodological*, 128(10):408–434, 2019.

[29] Douglas B. West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, NJ, 1996.

[30] Haohan Xiao and Meng Xu. How to restrain participants opt out in shared parking market? a fair recurrent double auction approach. *Transportation Research Part C: Emerging Technologies*, 93(8):36–61, 2018.

[31] Haohan Xiao, Meng Xu, and Ziyou Gao. Shared parking problem: A novel truthful double auction mechanism approach. *Transportation Research Part B: Methodological*, 109(3):40–69, 2018.

[32] Suxiu Xu, Meng Cheng, Xiang T.R. Kong, Hai Yang, and George Q. Huang. Private parking slot sharing. *Transportation Research Part B: Methodological*, 93(11):596–617, 2016.

[33] Hai Yang, Wei Liu, Xiaolei Wang, and Xiaoning Zhang. On the morning commute problem with bottleneck congestion and parking space con-

straints. *Transportation Research Part B: Methodological*, 58(12):106–118, 2013.

[34] Xiaoning Zhang, Hai Yang, and Hai Jun Huang. Improving travel efficiency by parking permits distribution and trading. *Transportation Research Part B: Methodological*, 45(7):1018–1034, 2011.

[35] Bo Zou, Nabin Kafle, Ouri Wolfson, and Jie(Jane) Lin. A mechanism design based approach to solving parking slot assignment in the information era. *Transportation Research Part B: Methodological*, 81(11):631–653, 2015.

Appendix A. Experiment setting

Appendix A.1. Description of collected data

We obtain a real data set with 1608 demands and 995 supplies in Xi'dan Central Business District (CBD) of Beijing from Oct. 1 to Dec. 31, 2015, provided by "DingdingParking" application. Each piece of the demand data records the announcement time when a driver sent her demand to the system, the reserved parking time and the actual arrival and departure times from the shared parking spaces if the driver was matched with a parking space. Each piece of the supply data records the announcement time when an owner sent his supply to the system, and the available time interval of the shared parking space. We artificially classify drivers into three types: Type I drivers are mostly commuters in early morning and types II and III drivers are visitors in later morning and afternoon. Also, we classify owners into three types: Type I owners are mostly commuters in early morning and Types II and III owners are irregular travelers with occasional travel purposes in later morning and afternoon. The average announcement rates and the average parking or supplying time of three type drivers and owners are listed in Tables A.3 and A.4, respectively.

Table A.3: Statistical descriptions of three type drivers

Types of driver	Announcement rates for every 10 *mins*	Average parking time (*mins*)
Type I during 7:00 am - 9:00 am	0.54	298
Type II during 9:00 am - 12:00 am	0.46	142
Type III during 2:00 pm - 6:00 am	0.43	99

Table A.4: Statistical descriptions of three type owners

Types of owners	Announcement rates for every 10 *mins*	Average supplying time (*mins*)
Type I during 6:00 am - 10:00 am	0.40	572
Type II during 10:00 am - 12:00 am	0.13	467
Type III during 1:00 pm - 3:00 pm	0.08	269

Appendix A.2. Generation methods of parameters

We establish an experimental simulation bed to generate the demands and supplies randomly announced from three types of drivers and owners through a time horizon from 6:00 am to 6:00 pm. The current system of "DingdingParking" application does not request drivers to report their earliest departure time (a_i), latest arrival time (b_i), or stay duration (e_i) to the system. We assume that these missing parameters are subjected to normal probability distributions and estimate them by the observable data. For example, we generate b_i of type I drivers by using the collected driver average actual arrival time at the shared parking slots added a random value draw from a normal distribution $N(x, \sigma)$, namely b_i= N(8:00 am, 10 min), where 8:00 am is the average arrival times and 10 min is the estimated standard variance, both of which are listed under the Arrival time (b_i) in Table A.5. Accordingly, the earliest departure time is generated as $a_i = b_i - t_i - \delta_i$, where δ_i denotes the slack time between b_i and a_i subtracted t_i, and is used as a controlling parameter in this experimen-

t. Similarly, stay durations at the destinations (e_i) of three type drivers, the lower bounds of available time intervals (h_j), and the length of the available time interval (i.e., Δ_j) are randomly generated by function $N(x, \sigma)$, where x and σ are set as the corresponding parameters of the three types of drivers and owners in Tables A.5 and A.6, respectively. The upper bound of the available time interval of parking space j is calculated as $k_j = h_j + \Delta_j$.

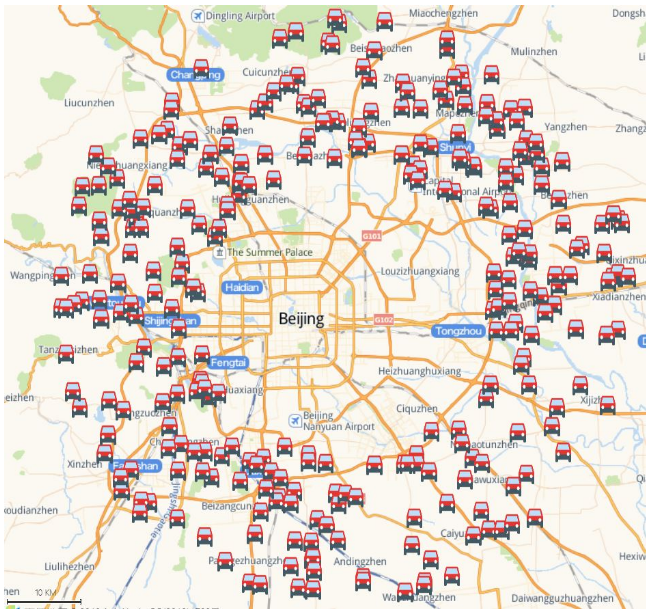
In this experiment, we use a function $LG(x_0, y_0, U(r_1, r_2), U(0°, 360°))$ to randomly generate the locations of origins (o_i) and destinations (d_i) of drivers and the locations of parking spaces. In the location-generation function LG, parameters x_0 and y_0 denote the longitude coordinates of the circle dot of the CBD, an uniform distribution function $U(r_1, r_2)$ randomly generates a distance between r_1 *km* and r_2 *km* of the location from the circle dot (x_0, y_0) and $U(0°, 360°)$ randomly generates an angle between 0° and 360° of the location. Hence, we can uniquely generate a location in the CBD with the randomly generated parameters: x_0, y_0, $U(r_1, r_2)$, and $U(0°, 360°)$. The locations of origins (o_i) and destinations (d_i) of drivers are generated by two functions $LG(x_0, y_0, U(20\ km, 40\ km), U(0°, 360°))$ and $LG(x_0, y_0, U(0\ km, 1\ km), U(0°, 360°))$, respectively. The locations (r_j) of parking spaces are randomly generated by $LG(x_0, y_0, U(0\ km, 1\ km), U(0°, 360°))$. Consequently, the origins and destinations of 300 drivers and locations of 200 parking spaces are illustrated in Figures A.10 (a)-(c), respectively.

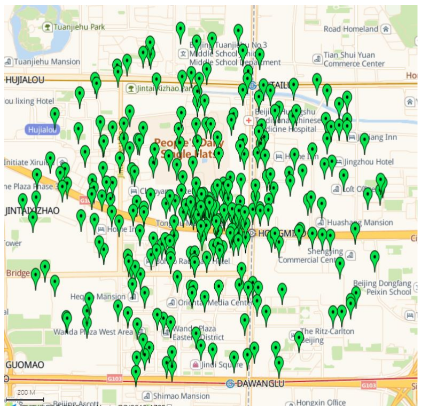
Table A.5: Parameters used to generate demands

Types of	Latest arrival time (b_i)		Stay duration (e_i)	
Drivers	mean	standard variance	mean	standard variance
type I	8:00 am	10 mins	5 hours	30 mins
type II	11:00 am	10 mins	2 hours	10 mins
type III	3:30 pm	10 mins	2 hours	10 mins

Table A.6: Parameters used to generate supplies

Types of owners	Lower bound of available time interval (h_j)		Length of available time interval (Δ_j)	
	mean	standard variance	mean	standard variance
type I	6:30 am	10 mins	12 hours	20 mins
type II	9:30 am	10 mins	10 hours	10 mins
type III	2:00 pm	10 mins	6 hours	10 mins


(a)Origins of drivers


(b) Destinations of drivers

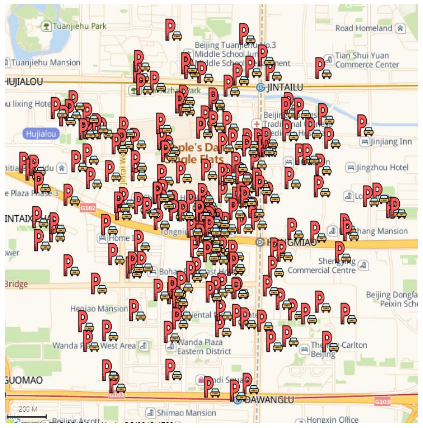

(c) Locations of shared parking spaces

Figure A.10: Locations of drivers' origins and destinations and parking spaces