Online scheduling of car-sharing request pairs between two locations

Kelin Luo¹ · Yinfeng Xu² · Haodong Liu²

Published online: 24 August 2020
© The Author(s) 2020

Abstract

We consider an online car-sharing problem between two locations with advance bookings. Each customer submits a pair of requests, where each request specifies the pick-up time and the pick-up location: one request from A to B , and the other request from B to A , not necessary in this order. The scheduler aims to maximize the number of satisfied customers, where the schedule has to decide whether or not to accept a request pair immediately at the time when the request pair is submitted. This problem is called **ONLINETRANSFERSFORCOMMUTING**. We present lower bounds on the competitive ratio for this problem with both fixed booking times and variable booking times, and propose two algorithms, greedy algorithm and balanced greedy algorithm, that achieve the best possible competitive ratios.

Keywords Car-sharing problem · Online scheduling · Competitive analysis · Request pairs

1 Introduction

The sharing economy has become more close to everyone. The car-sharing, as a new form of transport, has already been in use in daily life. In a car-sharing system, customers can book a car to drive from one place to another, and the car-sharing company will provide a car to serve this customer. Once a request for booking a car is accepted,

✉ Kelin Luo
k.luo@tue.nl

Yinfeng Xu
yfxu@xjtu.edu.cn

Haodong Liu
HaodongLiu0515@outlook.com

¹ Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

² School of Management, Xi'an Jiaotong University, Xi'an, China

the customer can pick up a car at the origin, drive it to the destination and return it there. An optimization problem is denoted by *car-sharing problem* ($CS - k$ in short), where the goal is to maximize the number of satisfied customers by k cars. A customer is satisfied if all his or her requests are fulfilled.

In this paper, we study an online car-sharing problem where each customer submits two rides in opposite directions. This variant of online car-sharing problem is denoted by *ONLINE TRANSFERS FOR COMMUTING* ($OTFC - k$ in short, where k is the number of cars), and the goal is to maximize the number of satisfied customers. A scenario is as follows: There are two popular locations, which could be a business center and a residential area. Customers want to drive from the business center to the residential area and drive back later and vice versa.

Remark This work expanded the work of Luo et al. (2019b): we introduce the fixed booking time variant and variable booking time variant for this problem; we propose a balanced greedy algorithm for $OTFC - k$ ($k > 2$) and prove that BGA is optimal.

1.1 Related work

OFF-LINE CAR-SHARING PROBLEM Böhmová et al. (2016) studied two off-line versions of the car-sharing problems. In one problem, *OFFLINE TRANSFERS FOR COMMUTING*, there are two locations and each customer submits two requests (each request is specified by its pick-up time, pick-up location and drop-off location) that are between these two locations (in opposite directions). A customer is satisfied if and only if his or her two requests are fulfilled. The goal is to maximize the number of satisfied customers. If all customer requests for car bookings are known in advance, they proved that this problem is NP-hard and APX-hard even for one car. Following their works, we consider the online version of the problem. In the other problem, each customer submits one request, they showed that this problem can be solved in polynomial time using a minimum-cost network flow algorithm for both two different goals: To minimize the cars to serve all requests, or to maximize the number of accepted requests with k cars.

Bei and Zhang (2018) studied another version of the car-sharing problem. There are n cars, and $2n$ requests and each request is specified only by pick-up location and drop-off location. Each car serves exactly two requests. The goal is to assign all requests to the cars with minimized total travel cost. They showed that this problem is NP-hard, and gave an 2.5-approximation algorithm.

ONLINE CAR-SHARING PROBLEM In the online model, each request is revealed when it is submitted. The scheduler has to decide whether or not to accept a request immediately when it is submitted. Luo et al. (2018a) considered the car-sharing problem for the special case of two locations and one car. For fixed booking time variant and variable booking time variant, they proposed an optimal algorithm that is 2-competitive for fixed booking times and 3-competitive for variable booking times, respectively. With the case of two locations and two cars, for fixed booking time variant, they proved tight results for the competitive ratio (Luo et al. 2018b). The competitive ratio was shown to be 2. With the case of two locations and k cars, for both fixed and variable booking times, they showed that the case with 3 cars for the fixed booking

time variant and that with 5 cars for the variable booking time variant already allow us to get the best competitive ratio, and no improvement is possible with more cars (Luo et al. 2018c). In this paper, we consider the online car-sharing problem where each customer submits a pair of requests. For the online car-sharing problem with one car, the competitive ratio of the problem in which each customer submits a pair of requests is larger than the problem in which each customer submits one request. If each customer can submit one request or two requests, the competitive ratios are the same as the problem where each customer submits a pair of requests.

Furthermore, they dealt with the setting with m locations in a star network and k cars which is motivated by the transport between an airport and the nearby hotels (Luo et al. 2019a). They studied two star networks. In the unit star network, they gave a 2-competitive algorithm. For the arbitrary star network where the travel time between the airport and a hotel may have an arbitrary length between t and Lt for $L \geq 1$, they gave a $O(\log L)$ -competitive algorithm. For both networks, they proved the matching lower bounds on the competitive ratio of any deterministic online algorithm. Furthermore, in a metric graph, for both fixed booking time variant and variable booking time variant, Liu et al. (2019) gave an algorithm with competitive ratio $O(L)$, where the L is known in advance. They also proved that no $O(L^\alpha)$ (any $0 < \alpha < 1$)-competitive deterministic algorithm exists for the fixed booking variant, and no $O(L)$ -competitive deterministic algorithm exists for the variable booking variant.

ONLINE MATCHING PROBLEM The results of the special online car-sharing problem also can be viewed as a matching. Recently, Dutta and Sholley (2018) proposed a model of matching requests online. The requests are released when they are submitted and at the same time the profit of matching with other preceding d requests are revealed. The scheduler has to match each request within w time steps since it arrives and the objective is to construct a matching to maximize the sum of weights of the matched requests picked in the matching. They presented a randomized $1/4$ -competitive algorithm for this problem with $w = d$. Raghvendra (2018) came up with the online bipartite matching problem for ride sharing. Given a set of car locations in a metric space, the requests arrive one by one, and the scheduler has to decide immediately to match it to a car at a cost (the distance between locations) with the aim of minimizing the total travel distance. They improved the analysis of the deterministic Robust Matching algorithm from $O(\log^2 n)$ (Nayyar and Raghvendra 2017) to $O(\log n)$ for the line metric, and established a bound of $O(\log n)$ on the competitive ratio.

ONLINE INTERVAL SCHEDULING PROBLEM For an online car-sharing problem, if all the pick-up and drop-off locations are the same, the problem becomes an online interval scheduling problem. Lipton (1994) considered the online interval scheduling problem, where the intervals are presented in order of their start times and each interval has a length. In the problem setting, the scheduler has to decide whether to accept each interval before the next interval is presented, and ensure that no pair of accepted intervals overlap. The objective of this problem is to maximize the total length of the accepted intervals. They showed that no algorithm can achieve competitive ratio smaller than $O(\log \Delta)$ (where Δ denotes the ratio of the longest interval to the shortest interval, and Δ is unknown to the algorithm), and gave a randomized algorithm with competitive ratio $O((\log \Delta)^{1+\varepsilon})$.

Another interval scheduling problem where two or multiple intervals have to be accepted together, is called *multitrack interval problem* (Golumbic 2004; Gyárfás and West 1995). Bar-Yehuda et al. (2006) considered a multitrack interval model, where each job consists at most two segments (or intervals). The object is to find a scheduling that maximize the number of accepted jobs. They presented a greedy algorithm with 3-competitive ratio for the proper 2-interval graph, and proved that no randomized algorithm can achieve a competitive ratio smaller than 2.5. For general instances, they gave a $O(\log^2 \Delta)$ -competitive (resp. $O((\log \Delta)^{2+\varepsilon})$ -competitive) randomized algorithm, where Δ denotes the known (resp. unknown) ratio of the longest segment to the shortest segment.

1.2 Preliminaries

We consider a setting with two locations, one is denoted by 0 and the other is denoted by 1. Assume that the traveling time from 0 to 1 is the same as that from 1 to 0, and is denoted by t . We have k cars $S = \{s_1, s_2, \dots, s_k\}$. All of these cars stay at location 0 initially.

Each customer submits one request pair, which consists of two requests, and the directions of these two requests are in opposite. Let R denote a sequence of request pairs that are released over time. For a given sequence R , the i -th request pair in R is denoted by $\{r_{i,1}, r_{i,2}\}$, where $r_{i,1} = (b_{i,1}, t_{i,1}, p_{i,1})$ and $r_{i,2} = (b_{i,2}, t_{i,2}, p_{i,2})$. Note that for any request pair $\{r_{i,1}, r_{i,2}\}$, the request $r_{i,1}$ and request $r_{i,2}$ are released at the same time, i.e., $b_{i,1} = b_{i,2}$. Request $r_{i,1}$ (resp. $r_{i,2}$) is specified by the booking time $b_{i,1}$ (resp. $b_{i,2}$), the start time/the pick-up time $t_{i,1}$ (resp. $t_{i,2}$) and the start location/the pick-up location $p_{i,1}$ (resp. $p_{i,2}$). Besides, the drop-off time of $r_{i,1}$ is denoted by $\dot{t}_{i,1}$, the drop-off location is denoted by $\dot{p}_{i,1}$, where $\dot{t}_{i,1} = t_{i,1} + t$ and $\dot{p}_{i,1} = 1 - p_{i,1}$. In our problem setting, each customer submits a round-trip, that means the pick-up location of the request $r_{i,2}$ and the drop-off location of the request $r_{i,1}$ are the same, i.e., $p_{i,2} = \dot{p}_{i,1}$. And the pick-up time of $r_{i,2}$ is not later than the drop-off time of $r_{i,1}$, i.e. $t_{i,2} \geq t_{i,1} + t$.

We consider both the fixed booking time variant and the variable booking time variant. For fixed booking time variant, customers can submit a request pair at a specific time before the start time of the first request. $t_{i,1} - b_{i,1} = a$ holds for all i . Correspondingly, for variable booking time variant, customers can submit a request pair at any time of an interval before the start time of the first request. $b_l \leq t_{i,1} - b_{i,1} \leq b_u$ with $b_u > b_l \geq t$ holds for all i .

Each car can only serve one request at a time. We say that two requests are *in conflict* if they can not be served by one car. Note that $t_{i,2} \geq t_{i,1} + t$ and $p_{i,2} = \dot{p}_{i,1}$ ensure that the two requests in one request pair are not in conflict. When a request pair is accepted, the car-sharing company will provide a car for this customer at time $t_{i,1}$ at location $p_{i,1}$, and later provide a car at time $t_{i,2}$ at location $p_{i,2}$. We do not require that the algorithm assigns the request to a car immediately or assigns the two requests in one pair to the same car, when the requests are accepted by the algorithm.

We forbid ‘unprompted’ empty moves, similar to Luo et al. (2018c). An empty move is to move a car from one location to another without serving a request. An

algorithms is allowed to make an empty move if and only if the empty move is done to serve a request at the other location, which is accepted before the current time.

The performance of an algorithm for $OTFC - k$ is measured using competitive analysis (Borodin and El-Yaniv 2005). For any sequence of request pairs R , let P_{R^*} denote the objective value produced by an optimal scheduler OPT , where OPT has full information about R in advance, and P_{R^A} denote the objective value produced by an online deterministic algorithm A . The competitive ratio of A is defined by ρ_A , which is shown by $\rho_A = \sup_R \frac{P_{R^*}}{P_{R^A}}$. An algorithm A is ρ -competitive if $P_{R^*} \leq \rho \cdot P_{R^A}$ holds for all sequences R . For a problem, we say β is the *lower bound* on the best possible competitive ratio if $\rho_A \geq \beta$ for all $A \in ON$, where ON is the set of all online deterministic algorithms. And we say an algorithm A is optimal if $\rho_A = \beta$.

1.3 Paper outline

An overview of our results for ONLINETRANSFERSFORCOMMUTING with one car is shown in Table 1. For $OTFC - 1$ with fixed booking times $a \geq t$, no deterministic online algorithm can achieve a competitive ratio smaller than 5, and for $OTFC - 1$ with variable booking times $b_u > b_l \geq t$, no deterministic online algorithm can achieve a competitive ratio smaller than 6. The Greedy Algorithm is optimal for both variants above. Compared with $CS - 1$ problem, the lower bounds and upper bounds for $OTFC - 1$ are larger.

An overview of our results for ONLINETRANSFERSFORCOMMUTING with k ($k \geq 2$) cars is shown in Table 2. The lower bound for $OTFC - 2$ is 6, and the greedy algorithm is proved to be optimal. The lower bound for $OTFC - k$ ($k > 2$) is 4. When the pick-up time $t_{i,j}$ is an integer multiple of the travel time t , i.e., $t_{ij} = v \cdot t$ for some $v \in \mathbb{N}$, we propose a balanced greedy algorithm for $OTFC - k$ ($k > 2$ and $k = 2i, i \in \mathbb{N}$) which is optimal. When $k \neq 2i, i \in \mathbb{N}$, the upper bound for $OTFC - k$ ($k > 2$) with slotted pick-up times is $4 + \frac{4}{k-1}$.

The remaining of the paper is organized as follows. In Sect. 2, we prove the lower bounds. In Sect. 3, we propose the greedy algorithm for $OTFC - k$ and prove the competitive ratios. Section 4 concludes the results of this work.

2 Lower bound

In this section, we present lower bounds for $OTFC - 1$, $OTFC - 2$ and $OTFC - k$ ($k > 2$), respectively. We use ALG to denote any online algorithm, and OPT to denote the optimal scheduler. For a sequence of request pairs R , let R^* denote the set of request pairs accepted by OPT , and R' denote the set of request pairs accepted by ALG .

To illustrate whether two request pairs can be served by the same car or not, we have the following Observation 1.

Observation 1 (*Conflicted condition*) Any two requests $r_{a,b}$ and $r_{g,h}$ are in conflict if $p_{a,b} = p_{g,h}$ and $|t_{a,b} - t_{g,h}| < t$ (Fig. 1a shows an example), or if $p_{a,b} = p_{g,h}$ and $|t_{a,b} - t_{g,h}| < 2t$ (Fig. 1b shows an example).

Table 1 Compare the competitive ratios between $OTFC - 1$ and $CS - 1$

Problem	LB $0 \leq a < t$	UB	LB $a \geq t$	UB	LB $b_u > b_l \geq t$	UB
$OTFC - 1$	∞ (Th.1)	–	5 (Th.2)	5 (Th.6)	6 (Th.3)	6 (Th.6)
$CS - 1$	1 (Th.2 Luo et al. 2018a)	1 (Th.2 Luo et al. 2018a)	2 (Th.1 Luo et al. 2018a)	2 (Th.3 Luo et al. 2018a)	3 (Th.5 Luo et al. 2018a)	3 (Th.9 Luo et al. 2018a)

Table 2 Lower and upper bounds on the competitive ratio for $OTFC - k$

Problem	LB	UB
$OTFC - 2$	6 (Th. 4)	6 (Th. 7)
$OTFC - k$	4 (Th. 5)	$4(k = 2i, i \in \mathbb{N})$ (Th. 8)

Fig. 1 Illustration of conflicted requests

Note that the two requests of a request pair are not in conflict, that means for any request pair $\{r_{i,1}, r_{i,2}\} \in R$, we have $t_{i,2} - t_{i,1} \geq t$.

2.1 ONLINETRANSFERSFORCOMMUTING with one car

In this section, we present lower bounds for $OTFC - 1$.

Theorem 1 For $0 \leq a < t$, no deterministic online algorithm for $OTFC - 1$ with fixed booking times can achieve a constant competitive ratio.

Proof Initially, the adversary releases $\{r_{1,1}, r_{1,2}\} = \{(0, a, 0), (0, 3t, 1)\}$. We distinguish two cases.

Case 1: ALG accepts request pair $\{r_{1,1}, r_{1,2}\}$. The adversary releases request pairs (Fig. 2 shows an example):

$$\begin{aligned} \{r_{2,1}, r_{2,2}\} &= \{(0, a, 0), (0, 6t, 1)\}, \\ \{r_{3,1}, r_{3,2}\} &= \{(2t - a, 2t, 1), (2t - a, 7t, 0)\}, \\ \{r_{4,1}, r_{4,2}\} &= \{(3t - a, 3t, 0), (3t - a, 8t, 1)\}, \\ \{r_{5,1}, r_{5,2}\} &= \{(4t - a, 4t, 1), (4t - a, 9t, 0)\}. \end{aligned}$$

OPT accepts all request pairs $\{r_{2,1}, r_{2,2}\}, \{r_{3,1}, r_{3,2}\}, \{r_{4,1}, r_{4,2}\}, \{r_{5,1}, r_{5,2}\}$. However, ALG cannot accept all these request pairs since $r_{2,1}$ and $r_{1,1}$ are in conflict, and $r_{3,1}$ (also $r_{4,1}, r_{5,1}$) and $r_{1,2}$ are in conflict.

Observe that the ALG car stays at location 0 after serving $r_{1,2}$ and the OPT car stays at location 1 after serving $r_{5,2}$. The adversary then releases request pairs $\{r_{i,1}, r_{i,2}\} = \{((2i - 2)t - a, (2i - 2)t, 1), ((2i - 2)t - a, (2i - 1)t, 0)\}$ for all $6 \leq i \leq G$ (G can be arbitrarily large). Since we forbid 'unprompted' moves and $a < t$, a car does not have enough time to reach the pick-up location of the first request if the car stays at another location, ALG cannot accept all request pairs $\{r_{i,1}, r_{i,2}\} = \{((2i - 2)t - a, (2i - 2)t, 1), ((2i - 2)t - a, (2i - 1)t, 0)\}$ with $6 \leq i \leq G$. However, OPT can accept request pairs $\{r_{i,1}, r_{i,2}\}$ for all $6 \leq i \leq G$ since the OPT car stays at location 1 on time $(2i - 2)t$ with $6 \leq i \leq G$.

Fig. 2 Illustration of R^* and R' in Case 1 (Color figure online)

Fig. 3 Illustration of R^* and R' in Case 1 (Color figure online)

From the above analysis we have $|R^*| = G - 1$ and $|R'| = 1$, and hence $\lim_{G \rightarrow \infty} \frac{|R^*|}{|R'|} = \infty$.

Case 2: *ALG* does not accept request pair $\{r_{1,1}, r_{1,2}\}$. In this case, *OPT* accepts $\{r_{1,1}, r_{1,2}\}$. We have $|R^*| = 1$ and $|R'| = 0$, and hence $\frac{|R^*|}{|R'|} = \infty$. $\square$

Theorem 2 For $a \geq t$, no deterministic online algorithm for *OTFC* – 1 with fixed booking times can achieve a competitive ratio smaller than 5.

Proof Without loss of generality, suppose $(i - 1)t < a \leq it$ with $i \in \mathbb{N}$. Initially, the adversary releases a request pair $\{r_{1,1}, r_{1,2}\} = \{(0, a, 0), (0, (i + 3)t, 1)\}$. We distinguish two cases.

Case 1: *ALG* accepts request pair $\{r_{1,1}, r_{1,2}\}$. The adversary releases request pairs (Fig. 3 shows an example):

$$\begin{aligned} \{r_{2,1}, r_{2,2}\} &= \{(0, a, 1), (0, (i + 6)t, 0)\}, \\ \{r_{3,1}, r_{3,2}\} &= \{((i + 1)t - a, (i + 1)t, 0), ((i + 1)t - a, (i + 7)t, 1)\}, \\ \{r_{4,1}, r_{4,2}\} &= \{((i + 2)t - a, (i + 2)t, 1), ((i + 2)t - a, (i + 8)t, 0)\}, \\ \{r_{5,1}, r_{5,2}\} &= \{((i + 3)t - a, (i + 3)t, 0), ((i + 3)t - a, (i + 9)t, 1)\}, \\ \{r_{6,1}, r_{6,2}\} &= \{((i + 4)t - a, (i + 4)t, 1), ((i + 4)t - a, (i + 10)t, 0)\}. \end{aligned}$$

OPT accepts all request pairs $\{r_{2,1}, r_{2,2}\}, \{r_{3,1}, r_{3,2}\}, \{r_{4,1}, r_{4,2}\}, \{r_{5,1}, r_{5,2}\}, \{r_{6,1}, r_{6,2}\}$. However, *ALG* cannot accept anyone of these request pairs since $r_{2,1}$ (also $r_{3,1}$) and $r_{1,1}$ are in conflict, and $r_{4,2}$ (also $r_{5,2}, r_{6,2}$) and $r_{1,2}$ are in conflict. We have $|R^*| = 5$ and $|R'| = 1$, and hence $\frac{|R^*|}{|R'|} = 5$.

Case 2: *ALG* does not accept request pair $\{r_{1,1}, r_{1,2}\}$. In this case, *OPT* accepts $\{r_{1,1}, r_{1,2}\}$. We have $|R^*| = 1$ and $|R'| = 0$, and hence $\frac{|R^*|}{|R'|} = \infty$. $\square$

Theorem 3 For $b_u > b_l \geq t$, no deterministic online algorithm for *OTFC* – 1 with variable booking times can achieve a competitive ratio smaller than 6.

Fig. 4 Illustration of R^* and R' in Case 1 (Color figure online)

Proof Initially, the adversary releases a request pair $\{r_{1,1}, r_{1,2}\} = \{(it - b_u, it, 0), (it - b_u, (i + 3)t, 1)\}$, where $(i - 1)t < b_u \leq it$, $i \in \mathbb{N}$. Let $\delta = \min\{b_u - b_l, t\}$, $\varepsilon < \delta$, and $\theta = b_u - \delta$. We distinguish two cases.

Case 1: ALG accepts request pair $\{r_{1,1}, r_{1,2}\}$. The adversary releases request pairs (Fig. 4 shows an example):

$$\begin{aligned} \{r_{2,1}, r_{2,2}\} &= \{(it - \varepsilon - \theta, it - \varepsilon, 0), (it - \varepsilon - \theta, (i + 11)t - \varepsilon, 1)\}, \\ \{r_{3,1}, r_{3,2}\} &= \{((i + 1)t - \varepsilon - \theta, (i + 1)t - \varepsilon, 1), ((i + 1)t - \varepsilon - \theta, (i + 10)t - \varepsilon, 0)\}, \\ \{r_{4,1}, r_{4,2}\} &= \{((i + 2)t - \varepsilon - \theta, (i + 2)t - \varepsilon, 0), ((i + 2)t - \varepsilon - \theta, (i + 9)t - \varepsilon, 1)\}, \\ \{r_{5,1}, r_{5,2}\} &= \{((i + 3)t - \varepsilon - \theta, (i + 3)t - \varepsilon, 1), ((i + 3)t - \varepsilon - \theta, (i + 8)t - \varepsilon, 0)\}, \\ \{r_{6,1}, r_{6,2}\} &= \{((i + 4)t - \varepsilon - \theta, (i + 4)t - \varepsilon, 0), ((i + 4)t - \varepsilon - \theta, (i + 7)t - \varepsilon, 1)\}, \\ \{r_{7,1}, r_{7,2}\} &= \{((i + 5)t - \varepsilon - \theta, (i + 5)t - \varepsilon, 1), ((i + 5)t - \varepsilon - \theta, (i + 6)t - \varepsilon, 0)\}. \end{aligned}$$

OPT accepts all request pairs $\{r_{2,1}, r_{2,2}\}, \{r_{3,1}, r_{3,2}\}, \{r_{4,1}, r_{4,2}\}, \{r_{5,1}, r_{5,2}\}, \{r_{6,1}, r_{6,2}\}, \{r_{7,1}, r_{7,2}\}$.

OPT accepts all request pairs $\{r_{2,1}, r_{2,2}\}, \{r_{3,1}, r_{3,2}\}, \{r_{4,1}, r_{4,2}\}, \{r_{5,1}, r_{5,2}\}, \{r_{6,1}, r_{6,2}\}, \{r_{7,1}, r_{7,2}\}$. However, ALG cannot accept anyone of these request pairs since $r_{2,1}$ (also $r_{3,1}, r_{4,1}$) and $r_{1,1}$ are in conflict, $r_{5,1}$ (also $r_{6,1}, r_{7,1}$) and $r_{1,2}$ are in conflict. We have $|R^*| = 6$ and $|R'| = 1$, and hence $\frac{|R^*|}{|R'|} = 6$.

Case 2: $\{r_{1,1}, r_{1,2}\}$ is not accept by ALG . The adversary stops releasing requests, and OPT accepts $\{r_{1,1}, r_{1,2}\}$ is this case. It shows that $|R^*| = 1$ and $|R'| = 0$, and hence $\frac{|R^*|}{|R'|} = \infty$. $\square$

2.2 ONLINE TRANSFERS FOR COMMUTING with two cars

In this section, we present lower bound for $OTFC - 2$. Recall that ALG denotes any online algorithm and OPT denotes the optimal scheduler, and R' and R^* denote the request pair set accepted by ALG and OPT , respectively. To show the lower bound, we introduce two conceptions: *matched* and *unmatched*.

Definition 1 For any two accepted requests $r_{a,b}$ and $r_{g,h}$, we say that $r_{a,b}$ and $r_{g,h}$ are matched if and only if they satisfy $|t_{a,b} - t_{g,h}| < 2t$ and $p_{a,b} = p_{g,h}$. Both $r_{a,b}$ and $r_{g,h}$ are called matched requests. If $r_{i,j}$ is not matched with any other accepted request, we say that $r_{i,j}$ is an unmatched request.

According to Observation 1, any two matched requests cannot be served by one car since they are in conflict.

Fig. 5 Illustration of two matched requests and six corresponding request pairs

Lemma 1 For *OTFC* – 2, if *ALG* accepts two matched requests $r_{a,b}$ and $r_{g,h}$ with $t_{a,b} \leq t_{g,h}$, then the adversary can release six request pairs (see Fig. 5 for an illustration):

$$\begin{aligned} \{r_{1,1}, r_{1,2}\} &= \{(b_0, t_{a,b} - \varepsilon, p_{a,b}), (b_0, t_{a,b} + 5t, 1 - p_{a,b})\}, \\ \{r_{2,1}, r_{2,2}\} &= \{(b_0, t_{a,b} - \varepsilon, p_{a,b}), (b_0, t_{a,b} + 5t, 1 - p_{a,b})\}, \\ \{r_{3,1}, r_{3,2}\} &= \{(b_0 + t, t_{a,b} + t - \varepsilon, 1 - p_{a,b}), (b_0 + t, t_{a,b} + 4t, p_{a,b})\}, \\ \{r_{4,1}, r_{4,2}\} &= \{(b_0 + t, t_{a,b} + t - \varepsilon, 1 - p_{a,b}), (b_0 + t, t_{a,b} + 4t, p_{a,b})\}, \\ \{r_{5,1}, r_{5,2}\} &= \{(b_0 + 2t, t_{a,b} + 2t - \varepsilon, p_{a,b}), (b_0 + 2t, t_{a,b} + 3t, 1 - p_{a,b})\}, \\ \{r_{6,1}, r_{6,2}\} &= \{(b_0 + 2t, t_{a,b} + 2t - \varepsilon, p_{a,b}), (b_0 + 2t, t_{a,b} + 3t, 1 - p_{a,b})\}, \end{aligned}$$

which can be accepted by *OPT* but can not be accepted by *ALG*, where $\varepsilon < \min\{t_{g,h} - t_{a,b}, t\}$, and $b_0 = \max\{b_{a,b}, b_{g,h}\}$.

Proof Observe that $r_{1,1}$ is in conflict with $r_{a,b}$ because $t_{a,b} - t_{1,1} < t$ (based on Observation 1). Similarly, $r_{2,1}$ (also $r_{3,1}, r_{4,1}, r_{5,1}, r_{6,1}$) is in conflict with $r_{a,b}$. Since $r_{a,b}$ and $r_{g,h}$ are matched by each other, by Definition 1 we have $t_{g,h} - t_{a,b} < 2t$. As $\varepsilon < \min\{t_{g,h} - t_{a,b}, t\}$, we have $t_{g,h} - t_{1,1} < t$, thus $r_{1,1}$ is in conflict with $r_{g,h}$ (based on Observation 1). Similarly, $r_{2,1}$ (also $r_{3,1}, r_{4,1}, r_{5,1}, r_{6,1}$) is in conflict with $r_{g,h}$. Therefore, *ALG* can not accept any of the six request pairs because *ALG* have already accepted $r_{a,b}$ and $r_{g,h}$.

Let s_1^* and s_2^* denote the two *OPT* cars. Since $p_{3,1} = 1 - p_{1,1} = \dot{p}_{1,1}$ and $t_{3,1} = t_{1,1} + t$, $r_{1,1}, r_{3,1}$ are not in conflict (based on Observation 1). Similarly, $r_{1,1}, r_{3,1}$, and $r_{5,1}$ are not in conflict, and $r_{1,2}, r_{3,2}$, and $r_{5,2}$ are not in conflict. Car s_1^* can serve all requests in $\{r_{1,1}, r_{1,2}\}, \{r_{3,1}, r_{3,2}\}, \{r_{4,1}, r_{4,2}\}$, and s_2^* can serve all other requests, the requests in $\{r_{2,1}, r_{2,2}\}, \{r_{5,1}, r_{5,2}\}, \{r_{6,1}, r_{6,2}\}$. This lemma is proved. $\square$

Now we give Theorem 4:

Theorem 4 No deterministic online algorithm for *OTFC* – 2 with variable bookings $b_u > b_l \geq t$ can achieve competitive ratio smaller than 6.

Proof Recall that R^* is the set of request pairs accepted by *OPT*, and R' is the set of request pairs accepted by *ALG*. Let R'' denote the union of the requests in R' .

At first, the adversary releases a request pair $\{r_{1,1}, r_{1,2}\} = \{((v+3)t - b_u, (v+3)t, 0), ((v+3)t - b_u, (v+15)t, 1)\}$, where $(v-1)t < b_u \leq v \cdot t$, $v \in \mathbb{N}$. If *ALG* rejects this request pair, then *OPT* accepts $\{r_{1,1}, r_{1,2}\}$, and it leads to $\frac{|R^*|}{|R'|} = \infty$.

Otherwise, $r_{1,1}$ and $r_{1,2}$ are accepted by *ALG* and they are unmatched (according to Definition 1). Let $r_{a,b}$ and $r_{g,h}$ be the unmatched requests in R'' . Note that $r_{a,b} = r_{1,1}$, $r_{g,h} = r_{1,2}$. Next the adversary releases request pairs according to the “If-else loop”, shown in Algorithm 1:

Algorithm 1 If-else Loop

```

1: Input:  $r_{a,b}, r_{g,h}$ .
2:  $i = 0$ ;
3: While  $i < 2$  do
4:   If  $i = 0$ ,
5:     the adversary releases  $\{r_{a_1,1}, r_{a_1,2}\}, \{r_{a_2,1}, r_{a_2,2}\}, \{r_{a_3,1}, r_{a_3,2}\}, \{r_{a_4,1}, r_{a_4,2}\}$ ;
6:     If ALG accepts one request pair  $\{r_{a_t,1}, r_{a_t,2}\}$ ,
7:       then request  $r_{a,b}$  and request  $r_{a_t,1}$  are matched. The adversary stops to release the rest request
       pairs, and releases 6 request
       pairs according to Lemma 1;
8:      $r_{a,b} = r_{g,h}, r_{g,h} = r_{a_t,2}$ ;
9:   Else
10:    OPT accepts  $\{r_{a_1,1}, r_{a_1,2}\}, \{r_{a_2,1}, r_{a_2,2}\}, \{r_{a_3,1}, r_{a_3,2}\}, \{r_{a_4,1}, r_{a_4,2}\}$ ;
11:     $i = i + 1$ ;
12:  Else if  $i = 1$ ,
13:    the adversary releases  $\{r_{b_1,1}, r_{b_1,2}\}, \{r_{b_2,1}, r_{b_2,2}\}, \{r_{b_3,1}, r_{b_3,2}\}, \{r_{b_4,1}, r_{b_4,2}\}$ ;
14:    If ALG accepts one request pair  $\{r_{b_t,1}, r_{b_t,2}\}$ ,
15:      then request  $r_{g,h}$  and request  $r_{b_t,1}$  are matched. The adversary stops to release the rest request
      pairs, and releases 6 request
      pairs according to Lemma 1;
16:       $r_{g,h} = r_{b_t,2}$ ;
17:    Else
18:      OPT accepts  $\{r_{b_1,1}, r_{b_1,2}\}, \{r_{b_2,1}, r_{b_2,2}\}, \{r_{b_3,1}, r_{b_3,2}\}, \{r_{b_4,1}, r_{b_4,2}\}$ ;
19:       $i = i + 1$ ;

```

$(1) \{r_{a_1,1}, r_{a_1,2}\} = (t_{a,b} - b_u + \varepsilon, t_{a,b} - 3\varepsilon, p_{a,b}), (t_{a,b} - b_u + \varepsilon, t_{g,h} + 8t, 1 - p_{a,b}),$
 $\{r_{a_2,1}, r_{a_2,2}\} = (t_{a,b} - b_u + 2\varepsilon, t_{a,b} - 2\varepsilon, p_{a,b}), (t_{a,b} - b_u + 2\varepsilon, t_{g,h} + 8t, 1 - p_{a,b}),$
 $\{r_{a_3,1}, r_{a_3,2}\} = (t_{a,b} + 2t - b_u, t_{a,b} + 2t - 3\varepsilon, p_{a,b}), (t_{a,b} + 2t - b_u, t_{g,h} + 10t, 1 - p_{a,b}),$
 $\{r_{a_4,1}, r_{a_4,2}\} = (t_{a,b} + 2t - b_u, t_{a,b} + 2t - 2\varepsilon, p_{a,b}), (t_{a,b} + 2t - b_u, t_{g,h} + 10t, 1 - p_{a,b}).$
 $(2) \{r_{b_1,1}, r_{b_1,2}\} = (t_{g,h} - b_u + \varepsilon, t_{g,h} - 3\varepsilon, p_{g,h}), (t_{g,h} - b_u + \varepsilon, t_{g,h} + 14t, 1 - p_{g,h}),$
 $\{r_{b_2,1}, r_{b_2,2}\} = (t_{g,h} - b_u + 2\varepsilon, t_{g,h} - 2\varepsilon, p_{g,h}), (t_{g,h} - b_u + 2\varepsilon, t_{g,h} + 14t, 1 - p_{g,h}),$
 $\{r_{b_3,1}, r_{b_3,2}\} = (t_{g,h} + 2t - b_u, t_{g,h} + 2t - 3\varepsilon, p_{g,h}), (t_{g,h} + 2t - b_u, t_{g,h} + 16t, 1 - p_{g,h}),$
 $\{r_{b_4,1}, r_{b_4,2}\} = (t_{g,h} + 2t - b_u, t_{g,h} + 2t - 2\varepsilon, p_{g,h}), (t_{g,h} + 2t - b_u, t_{g,h} + 16t, 1 - p_{g,h}).$

Then we analyze the number of request pairs accepted by *ALG* and *OPT*. In each loop, there are two cases:

Case 1: $i = 0$

The adversary releases four request pairs in order of the booking times:

$$\begin{aligned}
 \{r_{a_1,1}, r_{a_1,2}\} &= (t_{a,b} - b_u + \varepsilon, t_{a,b} - 3\varepsilon, p_{a,b}), (t_{a,b} - b_u + \varepsilon, t_{g,h} + 8t, 1 - p_{a,b}), \\
 \{r_{a_2,1}, r_{a_2,2}\} &= (t_{a,b} - b_u + 2\varepsilon, t_{a,b} - 2\varepsilon, p_{a,b}), (t_{a,b} - b_u + 2\varepsilon, t_{g,h} + 8t, 1 - p_{a,b}), \\
 \{r_{a_3,1}, r_{a_3,2}\} &= (t_{a,b} + 2t - b_u, t_{a,b} + 2t - 3\varepsilon, p_{a,b}), (t_{a,b} + 2t - b_u, t_{g,h} + 10t, 1 - p_{a,b}), \\
 \{r_{a_4,1}, r_{a_4,2}\} &= (t_{a,b} + 2t - b_u, t_{a,b} + 2t - 2\varepsilon, p_{a,b}), (t_{a,b} + 2t - b_u, t_{g,h} + 10t, 1 - p_{a,b}).
 \end{aligned}$$

Once *ALG* accepts a request pair $\{r_{a_t,1}, r_{a_t,2}\}$, the adversary stops to release the rest request pairs. Note that the request $r_{a,b}$ is matched by $r_{a_t,1}$, there are still two

unmatched requests in R'' : $r_{a,b} = r_{g,h}$ and $r_{g,h} = r_{a_i,2}$. The adversary releases six request pairs according to Lemma 1. OPT accepts all the six request pairs, while ALG can not accept either of these request pairs. In sum, ALG accepts one request pair and OPT accepts six request pairs in this loop (see Fig. 6 for an illustration).

Once the four request pairs are not accepted by ALG , $i = i + 1$ and the algorithm switch to Case 2. In sum, ALG accepts no request pairs and OPT accepts four request pairs in this loop (see red lines in Fig. 7 for an illustration). These four request pairs are counted when the loop break.

Case 2: $i = 1$.

The adversary releases four request pairs in order of the booking times:

$$\begin{aligned} \{r_{b_1,1}, r_{b_1,2}\} &= (t_{g,h} - b_u + \varepsilon, t_{g,h} - 3\varepsilon, p_{g,h}), (t_{g,h} - b_u + \varepsilon, t_{g,h} + 14t, 1 - p_{g,h}), \\ \{r_{b_2,1}, r_{b_2,2}\} &= (t_{g,h} - b_u + 2\varepsilon, t_{g,h} - 2\varepsilon, p_{g,h}), (t_{g,h} - b_u + 2\varepsilon, t_{g,h} + 14t, 1 - p_{g,h}), \\ \{r_{b_3,1}, r_{b_3,2}\} &= (t_{g,h} + 2t - b_u, t_{g,h} + 2t - 3\varepsilon, p_{g,h}), (t_{g,h} + 2t - b_u, t_{g,h} + 16t, 1 - p_{g,h}), \\ \{r_{b_4,1}, r_{b_4,2}\} &= (t_{g,h} + 2t - b_u, t_{g,h} + 2t - 2\varepsilon, p_{g,h}), (t_{g,h} + 2t - b_u, t_{g,h} + 16t, 1 - p_{g,h}). \end{aligned}$$

Similarly, once ALG accepts a request pair $\{r_{b_i,1}, r_{b_i,2}\}$, the adversary stops to release the rest request pairs. Note that the request $r_{g,h}$ is matched by $r_{b_i,1}$, there are still two unmatched requests in R'' : $r_{a,b}$ is unchanged and $r_{g,h} = r_{b_i,2}$. The adversary releases six request pairs according to Lemma 1. OPT accepts all the six request pairs, while ALG can not accept either of these request pairs. In sum, ALG accepts one request pair and OPT accepts six request pairs in this loop.

Once the four request pairs are not accepted by ALG , the loop breaks. In sum, ALG accepts no request pairs and OPT accepts four request pairs in this loop.

Next we will prove that any two requests accepted by OPT in two different loops are not in conflict. According to Algorithm 1, we have three circumstances: Both two loops x and y are in case 1, loop x in case 1 and loop y in case 2, or loops x, y are both in case 2.

For any two loops x and y ($x < y$) in case 1, we claim that $t_{a,b}$ in loop $x + 1$ is no less than $t_{a,b} + 8t$ in loop x . Observe that $t_{g,h} = t_{a,b} + 12t$ in loop 1. Based on Line 8 in Algorithm 1, we know that $t_{a,b}$ in loop $x + 1$ is equal to $t_{g,h}$ in loop x , and $t_{g,h}$ in loop $x + 1$ is equal to $t_{a_i,2}$ in loop x where $t_{a_i,2} = t_{g,h} + 8t$ or $t_{a_i,2} = t_{g,h} + 10t$. Thus we know that $t_{g,h} - t_{a,b} \geq 8t$ in loop $x + 1$, and $t_{a,b}$ in loop $x + 1$ is no less than $t_{a,b} + 8t$ in loop x , the claim is proved. For any two loops x and y ($x < y$) in case 2, we claim that $t_{g,h}$ in loop $x + 1$ is no less than $t_{g,h} + 14t$ in loop x . Based on Line 16 in Algorithm 1, we know that $t_{g,h}$ in loop $x + 1$ is equal to $t_{b_i,2}$ in loop x , which is later than $t_{g,h} + 14t$ in loop x . The above claim is proved.

Consider the circumstance that loop x and loop y are both in case 1. Since loop x and loop y are both in case 1, the requests accepted by OPT in loop x must have start time between $(t_{a,b} - t, t_{a,b} + 5t]$ based on Lemma 1. If OPT accepts 6 pair pairs in loop y as shown in Line 7 in Algorithm 1, we know that the start time of the request accepted by OPT in loop y is not earlier than $t_{a,b} - t$ in loop y , which is not earlier than $2t$ plus the start time of the request accepted by OPT in loop x with the latest start time since $t_{a,b}$ in loop y is no less than $t_{a,b} + 8t$ in loop x . Similarly, if OPT accepts 4 request pairs in loop y as shown in Line 10 in Algorithm 1, we know that

the start time of the request accepted by OPT in loop y is not earlier than $t_{a,b} - t$ in loop y , which is not earlier than $2t$ plus the start time of the request accepted by OPT in loop x with the latest start time. That means any two requests accepted by OPT in two different loop x and loop y are not in conflict.

Consider the circumstance that loop x and loop y are both in case 2. Since loop x and loop y are both in case 2, the requests accepted by OPT in loop x must have start time between $(t_{g,h} - t, t_{g,h} + 5t]$ based on Lemma 1. If OPT accepts 6 request pairs in loop y as shown in Line 15 in Algorithm 1, we know that the start time of the request accepted by OPT in loop y is not earlier than $t_{g,h} - t$ in loop y , which is not earlier than $2t$ plus the start time of the request accepted by OPT in loop x with the latest start time since $t_{g,h}$ in loop y is no less than $t_{g,h} + 14t$ in loop x . Similarly, if OPT accepts 4 request pairs in loop y as shown in Line 18 in Algorithm 1, we know that the start time of the request accepted by OPT in loop y is not earlier than $t_{g,h} - t$ in loop y , which is not earlier than $2t$ plus the start time of the request accepted by OPT in loop x with the latest start time. That means any two requests accepted by OPT in two different loop x and loop y are not in conflict.

Consider the circumstance that loop x is the last case 1 loop and loop y is in case 2. As shown in Line 10 in Algorithm 1, the requests accepted in loop x starts between $(t_{a,b} - t, t_{a,b} + 2t]$ or $t_{g,h} + 8t$, or $t_{g,h} + 10t$. If y is the first case 2 loop, $y = x + 1$, $t_{a,b}$ (resp. $t_{g,h}$) in both loops are the same. As shown in Line 15 and Line 18 in Algorithm 1, the requests accepted in loop y starts between $(t_{g,h} - t, t_{g,h} + 5t]$ or $t_{g,h} + 14t$, or $t_{g,h} + 16t$. Since $t_{g,h} - t_{a,b} \geq 8t$, we know that any request accepted in loop x and a request accepted in loop y are not in conflict as the interval of the two start times is at least $2t$. If $y \geq x + 2$, $t_{g,h}$ in loop y is greater than $t_{g,h} + 14t$ in loop x . As shown in Line 15 and Line 18 in Algorithm 1, the requests accepted in loop y starts between $(t_{g,h} - t, t_{g,h} + 5t]$ or $t_{g,h} + 14t$, or $t_{g,h} + 16t$. Since $t_{g,h} - t_{a,b} \geq 8t$ in loop x , we know that any request accepted in loop x and a request accepted in loop y are not in conflict as the interval of the two start times is at least $3t$. That means any two requests accepted by OPT in the last case 1 loop x and case 2 loop y are not in conflict.

Since all requests accepted in case 1 by OPT starts earlier than the requests in the last case 1 loop, we deduce that any two requests accepted by OPT in two different loops are not in conflict based on the above three circumstances analysis.

Finally, we bound the accepted requests by ALG and OPT . If the loop finally breaks, based on the analysis of Case 1 and Case 2, in each loop that ALG accepts one request pair, OPT accepts six request pairs that ALG could not accept; and whenever ALG accepts no request pairs, OPT accepts four request pairs. Notice that ALG accepts no request pairs in exactly two loops and the loop breaks. Suppose ALG accepts n request pairs (one is $\{r_{1,1}, r_{1,2}\}$), then OPT accepts $6(n - 1) + 8$ request pairs. Therefore, we have $\frac{|R^*|}{|R'|} = \frac{6i+8}{i+1} > 6$.

If the loop does not break and works internally, based on the analysis of Case 1 and Case 2, for each loop that ALG accepts one request pair, OPT accepts six request pairs that ALG could not accept. Suppose ALG accepts n request pairs (one is $\{r_{1,1}, r_{1,2}\}$), then OPT accepts no less than $6(n - 1)$ request pairs. Since $n \rightarrow \infty$, we have $\frac{|R^*|}{|R'|} = \frac{6(n-1)}{n} = 6$. $\square$

Fig. 6 Illustration of requests accepted by *OPT* and *ALG* (Color figure online)

Fig. 7 Illustration of requests accepted by *OPT* and *ALG* (Color figure online)

2.3 ONLINE TRANSFERS FOR COMMUTING with k cars

Theorem 5 For $b_u > b_l \geq t$ and $b_u \geq b_l + t$, no deterministic online algorithm for *OTFC* – k with variable booking times can achieve a competitive ratio smaller than 4.

Proof Without loss of generality, suppose $t = 1$. Consider a sequence of request pairs that consists of 8 phases where phase i , $1 \leq i \leq 8$, consists of l_i groups of request pairs. $l_i = 1$ for $i = 1, 4, 6, 8$. Let $R_{i,j}$ denote the set of request pairs in phase i group j . Each group consists of k copies of a request pair. If $l_i = 1$, let $\{r_{i,1}, r_{i,2}\}$ denote the request pair in $R_{i,1}$; otherwise, let $\{r_{ij,1}, r_{ij,2}\}$ denote the request pair in $R_{i,j}$. Suppose *ALG* accepts $k_{i,j}$ request pairs in phase i group j , $R_{i,j}$. Let k_i denote the number of request pairs that *ALG* accepted in phase i , $\bigcup_j R_{i,j}$. Note that $k_i = \sum_j k_{i,j}$.

At first, the adversary releases $R_{1,1}$, consisting of k copies of the request pair $\{r_{1,1}, r_{1,2}\} = \{(0, b_u, 0), (0, b_u + 4, 1)\}$. We distinguish two cases based on the value of $k_{1,1}$.

Case 1: *ALG* accepts no more than $\frac{k}{4}$ request pairs in phase 1, i.e., $k_1 = k_{1,1} \leq \frac{k}{4}$. *OPT* accepts all the request pairs in $R_{1,1}$. We have $P_{R^*} = k$ and $P_{R'} \leq \frac{k}{4}$, and hence $\frac{P_{R^*}}{P_{R'}} \geq 4$.

Case 2: *ALG* accepts more than $\frac{k}{4}$ request pairs in phase 1, i.e., $k_1 > \frac{k}{4}$.

Then the adversary releases request pairs in phase 2 and phase 3 based on the releasing rule Algorithm 2. For all $1 \leq j \leq k$, $R_{2,j}$ consists of k copies of request pair $\{r_{2j,1}, r_{2j,2}\} = \{(0, b_u - \frac{j}{k+1}, 0), (0, 4 + b_u - \frac{j}{k+1}, 1)\}$; $R_{3,j}$ consists of k copies of request pair $\{r_{3j,1}, r_{3j,2}\} = \{(t_{3j,1} - b_u, t_{3j,1}, 0), (t_{3j,1} - b_u, 6 + b_u - \frac{l_2}{k+1} + \frac{j}{(k+1)^2}, 1)\}$, where $t_{3j,1} = 2 + b_u - \frac{l_2}{k+1} + \frac{j}{(k+1)^2}$.

Observe that $r_{2j,1}$ with $1 \leq j \leq l_2$ and $r_{1,1}$ are in conflict because $t_{1,1} - t_{2j,1} < 1$, and $r_{3j,1}$ with $1 \leq j \leq l_3$ and $r_{1,1}$ are in conflict because $t_{3j,1} - t_{1,1} < 2$ and

Algorithm 2 Releasing rule for phase i

```

1: Input:  $j = 1$  and the adversary releases  $R_{i,j}$ .
2: For  $k_{i,j} > 0$  do
3:    $j = j + 1$  and the adversary releases  $R_{i,j}$ ;
4: end for
5: Output:  $l_i = j, k_{i,j}$  ( $1 \leq j \leq l_i$ ).

```

$p_{3j,1} = p_{1,1}$, and $r_{2j,1}$ with $1 \leq j_1 < l_2$ and $r_{3j,1}$ with $1 \leq j_2 \leq l_3$ are in conflict because $t_{3j,1} - t_{1,1} < 2$ and $p_{2j,1} = p_{2j,1}$ (Based on Observation 1). Similarly, $r_{2j,2}$ with $1 \leq j \leq l_2$ and $r_{1,2}$ are in conflict, and $r_{3j,2}$ and $r_{1,2}$ are in conflict, and $r_{2j,2}$ with $1 \leq j_1 < l_2$ and $r_{3j,2}$ with $1 \leq j_2 \leq l_3$ are in conflict. According to Line 3 in Algorithm 2, we know that $k_{i,l_i} = 0$. Thus all request pairs accepted by *ALG* in the first three phases are in conflict. Observe that $r_{2l,1}$ and $r_{3l,1}$ are not in conflict because $t_{3l,1} - t_{2l,1} \geq 2$ (Based on Observation 1). Similarly, $r_{2l,2}$ and $r_{3l,1}$ are not in conflict, $r_{2l,2}$ and $r_{3l,2}$ are not in conflict.

We distinguish two cases based on the value of $k_1 + k_2 + k_3$.

Case 2.1: *ALG* accepts no more than $\frac{k}{2}$ request pairs in first three phases (phase 1, phase 2 and phase 3), i.e., $k_1 + k_2 + k_3 \leq \frac{k}{2}$.

Since $r_{2l,1}$ and $r_{3l,1}$ (also $r_{2l,2}$ and $r_{3l,2}$) are not in conflict, *OPT* can accept all request pairs in group R_{2,l_2} and R_{3,l_3} (see Fig. 8 for an illustration). We have $|R^*| = 2k$ and $|R'| = k_1 + k_2 + k_3 \leq \frac{k}{2}$, and hence $\frac{|R^*|}{|R'|} \geq 4$.

Case 2.2: *ALG* accepts more than $\frac{k}{2}$ request pairs in first three phases, i.e., $k_1 + k_2 + k_3 > \frac{k}{2}$.

The adversary then releases $R_{4,1}$, k copies of request pair $\{r_{4,1}, r_{4,2}\} = \{(b_{4,1}, 1 + b_u - \frac{l_2}{k+1} + \frac{l_3}{(k+1)^2}, 1), (b_{4,1}, 5 + b_u - \frac{l_2}{k+1} + \frac{l_3}{(k+1)^2}, 0)\}$, where $b_{4,1} = 2 - \frac{l_2}{k+1} + \frac{l_3}{(k+1)^2}$.

Observe that $r_{4,1}$ and $r_{1,1}$ are in conflict because $t_{4,1} - t_{1,1} < 1$, $r_{4,1}$ and $r_{2j,1}$ with $1 \leq j < l_2$ are in conflict because $t_{4,1} - t_{2j,1} < 1$, and $r_{4,1}$ and $r_{3,j}$ with $1 \leq j < l_3$ are in conflict because $t_{3j,1} - t_{4,1} < 1$ and (Based on Observation 1). Similarly, $r_{4,2}$ and $r_{1,2}$ are in conflict, $r_{4,2}$ and $r_{2j,2}$ with $1 \leq j < l_2$ are in conflict, and $r_{4,2}$ and $r_{3j,2}$ with $1 \leq j < l_3$ are in conflict. Since $k_{2,l_2} = 0$ and $k_{3,l_3} = 0$ (based on the releasing rule Algorithm 2), we know that before the adversary releases request pairs in phase 4, *ALG* only accepts request pairs in $R_{1,1}$, and $\bigcup_{j=1}^{l_2-1} R_{2,j}$, and $\bigcup_{j=1}^{l_3-1} R_{3,j}$. Note that all request pairs accepted by *ALG* in the first three phases are in conflict. From the above analysis we know that all request pairs accepted by *ALG* in the first four phases are in conflict, i.e., the $k_1 + k_2 + k_3 + k_4$ request pairs accepted by *ALG* are in conflict.

Observe that $r_{2l,1}$ and $r_{4,1}$ are not in conflict because $t_{4,1} - t_{2l,1} > 1$ and $p_{2l,1} = p_{4,1}$, and $r_{3l,1}$ and $r_{4,1}$ are not in conflict because $t_{3l,1} - t_{4,1} > 1$ and $p_{3l,1} = p_{4,1}$ (Based on Observation 1). Similarly, $r_{2l,2}$ and $r_{4,1}$ are not in conflict, $r_{2l,2}$ and $r_{4,2}$ are not in conflict, $r_{3l,2}$ and $r_{4,1}$ are not in conflict, and $r_{3l,2}$ and $r_{4,2}$ are not in conflict.

We distinguish two sub-cases based on the value of $k_1 + k_2 + k_3 + k_4$.

Case 2.2.1: *ALG* accepts no more than $\frac{3k}{4}$ request pairs in the first four phases, i.e., $k_1 + k_2 + k_3 + k_4 \leq \frac{3k}{4}$. Recall that the request pairs in R_{2,l_2} , R_{3,l_3} and $R_{4,1}$ are not in

conflict, that means OPT can accept all the request pairs in R_{2,l_2} , R_{3,l_3} and $R_{4,1}$ (see Fig. 9 for an illustration). We have $|R^*| = 3k$ and $|R'| \leq \frac{3k}{4}$, and hence $\frac{|R^*|}{|R'|} \geq 4$.

Case 2.2.2: ALG accepts more than $\frac{3k}{4}$ request pairs in the first four phases, i.e., $k_1 + k_2 + k_3 + k_4 > \frac{3k}{4}$. In other words, $k_1 + k_2 + k_3 + k_4$ cars are occupied by the request pairs in the first four phases.

Then the adversary releases request pairs in phase 5 based on the releasing rule Algorithm 2 and we have $k_{5,l_5} = 0$. $R_{5,j}$ consists of k copies of request pair $\{r_{5j,1}, r_{5j,2}\} = \{(t_{5j,1} - b_u + 1, t_{5j,1}, 1), (t_{5j,1} - b_u, 9 + b_u, 0)\}$, where $t_{5j,1} = 1 + b_u - \frac{l_2}{k+1} + \frac{l_3}{(k+1)^2} - \frac{j}{(k+1)^3}$. Next, the adversary releases request pairs in phase 6, $R_{6,1}$ consists of k copies of request pair $\{r_{6,1}, r_{6,2}\} = \{(t_{6,1} - b_u, t_{6,1}, 0), (t_{6,1} - b_u, 10 + b_u, 1)\}$, where $t_{6,1} = 2 + b_u - \frac{l_2}{k+1} + \frac{l_3}{(k+1)^2} - \frac{l_5}{(k+1)^3}$. Similar to the previous conflict request pair analysis, it is easy to show that $r_{5j,1}$ with $1 \leq j < l_5$ and $r_{1,1}$ (also $r_{2j,1}$ with $1 \leq j < l_2$, $r_{3j,1}$ with $1 \leq j < l_3$, and $r_{4,1}$) are in conflict, and $r_{6,1}$ and $r_{1,1}$ (also $r_{2j,1}$ with $1 \leq j < l_2$, $r_{3j,1}$ with $1 \leq j < l_3$, $r_{4,1}$, and $r_{5j,1}$ with $1 \leq j < l_5$) are in conflict. Note that $\{r_{2l_2,1}, r_{2l_2,2}\}$, $\{r_{5l_5,1}, r_{5l_5,2}\}$ and $\{r_{6,1}, r_{6,2}\}$ are not in conflict (see Fig. 10 for an illustration).

Furthermore, the adversary releases request pairs in phase 7 based on the releasing rule Algorithm 2 and we have $k_{7,l_7} = 0$. $R_{7,j}$ consists of k copies of request pair $\{r_{7j,1}, r_{7j,2}\} = \{(t_{7j,1} - b_u, t_{7j,1}, 0), (t_{7j,1} - b_u, 11 + b_u, 1)\}$, where $t_{7j,1} = 5 + b_u - \frac{l_2}{k+1} + \frac{l_3}{(k+1)^2} - \frac{j}{(k+1)^3}$. Finally, the adversary releases request pairs in phase 8, $R_{8,1}$ consists of k copies of request pair $\{r_{8,1}, r_{8,2}\} = \{(t_{8,1} - b_u, t_{8,1}, 1), (t_{8,1} - b_u, 12 + b_u, 0)\}$, where $t_{8,1} = 6 + b_u - \frac{l_2}{k+1} + \frac{l_3}{(k+1)^2} - \frac{l_7}{(k+1)^3}$. Similar to the previous conflict request pair analysis, it is easy to show that $r_{7j,1}$ with $1 \leq j < l_7$ and $r_{1,2}$ (also $r_{2j,2}$ with $1 \leq j < l_2$, $r_{3j,2}$ with $1 \leq j < l_3$, and $r_{4,2}$) are in conflict, and $r_{8,1}$ and $r_{1,2}$ (also $r_{2j,2}$ with $1 \leq j < l_2$, $r_{3j,2}$ with $1 \leq j < l_3$, $r_{4,2}$, and $r_{7j,1}$ with $1 \leq j < l_7$) are in conflict. Note that request pair $\{r_{2l_2,1}, r_{2l_2,2}\}$, $\{r_{5l_5,1}, r_{5l_5,2}\}$, $\{r_{6,1}, r_{6,2}\}$, $\{r_{7l_7,1}, r_{7l_7,2}\}$ and $\{r_{8,1}, r_{8,2}\}$ are not in conflict.

Since $\{r_{2l_2,1}, r_{2l_2,2}\}$, $\{r_{5l_5,1}, r_{5l_5,2}\}$, $\{r_{6,1}, r_{6,2}\}$, $\{r_{7l_7,1}, r_{7l_7,2}\}$ and $\{r_{8,1}, r_{8,2}\}$ are not in conflict, OPT accepts all the request pairs in R_{2,l_2} , R_{5,l_5} , $R_{6,1}$, R_{7,l_7} and $R_{8,1}$. We have $|R^*| = 5k$.

Since every request pair accepted by ALG in the first four phases is in conflict with $r_{5j,1}$ with $1 \leq j < l_5$ (also $r_{6,1}$, $r_{7j,1}$ with $1 \leq j < l_7$, and $r_{8,1}$), ALG at most use $k - k_1 - k_2 - k_3 - k_4$ cars to accept request pairs in the rest four phases. Furthermore, since every request pair accepted by ALG in the phase 6 is in conflict with $r_{5j,1}$ with $1 \leq j < l_5$, ALG at most accepts $k - k_1 - k_2 - k_3 - k_4$ requests in phase 5 and phase 6. Similarly, ALG at most accepts $k - k_1 - k_2 - k_3 - k_4$ requests in phase 7 and phase 8. Recall that $k_1 + k_2 + k_3 + k_4 > \frac{3k}{4}$, we have $k_5 + k_6 < \frac{k}{4}$ and $k_7 + k_8 < \frac{k}{4}$.

We have $|R'| \leq \frac{3k}{4} + \frac{k}{4} \times 2 = \frac{5k}{4}$, and hence $\frac{|R^*|}{|R'|} \geq 4$. $\square$

Fig. 8 Illustration of R^* and R' in Case 2.1 (Color figure online)

Fig. 9 Illustration of R^* and R' in Case 2.2.1 (Color figure online)

Fig. 10 Illustration of $R_{5,j}$ and $R_{6,j}$

3 Upper bound

In this section, we construct two algorithms to present upper bounds for $OTFC - 1$, $OTFC - 2$ and $OTFC - k$ ($k > 2$), respectively. Denote the request pairs accepted by OPT by $R^* = \{\{r_{1,1}^*, r_{1,2}^*\}, \{r_{2,1}^*, r_{2,2}^*\}, \dots, \{r_{|R^*|,1}^*, r_{|R^*|,2}^*\}\}$ and denote the request pairs accepted by an algorithm by $R' = \{\{r'_{1,1}, r'_{1,2}\}, \{r'_{2,1}, r'_{2,2}\}, \dots, \{r'_{|R'|,1}, r'_{|R'|,2}\}\}$, indexed in order of the booking times.

3.1 Greedy algorithm

In this section, we propose a Greedy algorithm (GA) for $ONLINE TRANSFERS FOR COMMUTING$, shown in Algorithm 3.

At first, we analysis the performance of GA for $OTFC - 1$ with fixed booking times $a \geq t$. Obviously, in the fixed booking times, the requests in R^* and R' are indexed

Algorithm 3 Greedy algorithm (GA)

- 1: *Input*: k cars, request pairs arrive over time.
- 2: When request pair $\{r_{i,1}, r_{i,2}\}$ arrives,
- 3: If $\{r_{i,1}, r_{i,2}\}$ is acceptable,
- 4: accept $\{r_{i,1}, r_{i,2}\}$;
- 5: Otherwise, reject it.

* Request pair $\{r_{i,1}, r_{i,2}\}$ is acceptable to a car if and only if there is a schedule that no more than k cars can serve all request pairs $R' \cup \{r_{i,1}, r_{i,2}\}$ without conflict.

Fig. 11 Illustration of later conflicted request pairs

in order of the pick-up times of the first requests. To analyze the competitive ratio for $OTFC - 1$ with fixed booking times $a \geq t$, we partition the request pairs in R^* into a number of request pair sets based on the following rules shown in Algorithm 4.

Algorithm 4 Partition Rules for $OTFC - 1$ with fixed booking times

- 1: *Input*: R' and R^* .
- 2: $i = 1$;
- 3: While $i \leq |R'|$ do
- 4: Let R_i^* be the set of request pairs that are conflict with $\{r'_{i,1}, r'_{i,2}\}$ and are booked at or later than $b_{i,1}$.
- 5: *Output*: R_i^*

Observation that the partition rule is complete. In other words, $R^* = \bigcup_{i=1}^{|R'|} R_i^*$. Otherwise, the request pairs that are not included in $\bigcup_{i=1}^{|R'|} R_i^*$ will be accepted by GA in *Step 1* in Algorithm 5.

Theorem 6 For $a \geq t$, GA is 5-competitive for $OTFC - 1$ with fixed booking times.

Proof We bound the competitive ratio of GA by analyzing each set R_i^* (obtained by the partition rule in Algorithm 4) independently. As $R^* = \bigcup_{i=1}^{|R'|} R_i^*$, it is clear that for any $\alpha \geq 1$, $|R^*|/|R'| \leq \alpha$ follows if we can show that $|R_i^*| \leq \alpha$ for all i , $1 \leq i \leq |R'|$.

Observe that for a request pair $\{r_{i,1}, r_{i,2}\}$, there are at most five request pairs whose first request starts later than $t_{i,1}$, which are in conflict with $\{r_{i,1}, r_{i,2}\}$ (see Fig. 11 shows an example). Thus $\alpha = 5$ and we have $|R_i^*| \leq 5$, and the theorem follows. $\square$

Now we analyze the competitive ratio of $OTFC - k$ with $k \geq 1$ for both fixed booking time variant and variable booking time variant. Without loss of generality, let R' denote the request pairs accepted by GA exactly before a new request pair arrives. Denote the union of the accepted requests in R' and the requests in $\{r_{i,1}, r_{i,2}\}$ by $R'' = \{r'_1, r'_2, \dots, r'_{2|R'|+2}\}$ indexed in order of non-decreasing request start times. The requests with same start time are ordered according to their releasing times.

We say that a request pair $\{r_{i,1}, r_{i,2}\}$ is acceptable if and only if there is a schedule that no more than k cars can serve all request pairs $R' \cup \{r_{i,1}, r_{i,2}\}$ without conflict. To determine whether a request pair is acceptable or not immediately, we have the following property, see Lemma 2.

Lemma 2 *When a request pair $\{r_{i,1}, r_{i,2}\}$ arrives, $\{r_{i,1}, r_{i,2}\}$ is acceptable if and only if the size of the maximum subset Sub is not greater than k , where Sub is a subset of R'' that any two requests in Sub are in conflict.*

Proof Recall that a request pair $\{r_{i,1}, r_{i,2}\}$ is acceptable if and only if there is a schedule that no more than k cars can serve all request pairs in $R' \cup \{r_{i,1}, r_{i,2}\}$ without conflict. *Sufficient condition* We claim that if the size of the maximum subset Sub is not greater than k , then there is a schedule that no more than k cars can serve all requests in R'' without conflict. If this claim holds, we get that $\{r_{i,1}, r_{i,2}\}$ is acceptable, proving the lemma. Suppose the maximum subset of R'' that any two requests in this subset are in conflict is R''_{sub} . Note that $|R''_{sub}| \leq k$. We schedule all requests in R'' one by one in order of request start times. For the simplification of analysis, we suppose that the cars are ordered. If a request $r_{i,j}$ is acceptable to some cars ($r_{i,j}$ and every request that assigned to these cars are not in conflict), we assign $r_{i,j}$ to the available car that comes first in that order. Since $|R''_{sub}| \leq k$, no more than k consecutive requests are assigned to k different cars. There always exists a car whose every accepted request and a new request are not in conflict, and hence we can always schedule a request in R'' to that car. This establishes the claim, proving the sufficient condition.

Necessary condition If $\{r_{i,1}, r_{i,2}\}$ is acceptable, then there is a schedule that no more than k cars can serve all request pairs $R' \cup \{r_{i,1}, r_{i,2}\}$ without conflict. Therefore the size of the maximum subset Sub is not greater than k . Otherwise, we get a contradiction because k cars could not serve all requests in subset Sub without conflict. $\square$

We refer to the car set of GA as $S' = \{s'_1, s'_2, \dots, s'_k\}$, and the car set of OPT as $S^* = \{s^*_1, s^*_2, \dots, s^*_k\}$. Let OPT_j be the request pairs accepted by car s^*_j , and let GA_j be the request pairs accepted by car s'_j .

Theorem 7 *For $b_u > b_l \geq t$, GA is 6-competitive for OTFC- k ($k \geq 1$) with variable booking times. And for $a \geq t$, GA is 6-competitive for OTFC- k ($k \geq 2$) with fixed booking times.*

Proof For car s^*_j , let $\bar{R}^*_j$ be the set of request pairs accepted by s^*_j that are not in R' , and let R'_j be the set of request pairs accepted by s'_j that are not in R^* . If the claim that $|\bigcup_j \bar{R}^*_j| \leq \alpha |\bigcup_k R'_j|$ for constant α holds and $\alpha = 6$, we have $\frac{|R^*|}{|R'|} \leq 6$, proving the lemma.

It remains to prove the claim. Consider any request pair $\{r_{h,1}, r_{h,2}\}$ in $\bar{R}^*_j$. As s'_j for all $s'_j \in S'$ did not accept request pair $\{r_{h,1}, r_{h,2}\}$, according to Algorithm 3 and Lemma 2, s'_j must have been scheduled another request pair $\{r_{c,1}, r_{c,2}\}$ in R'_j where one of the two requests is in conflict with request $r_{h,1}$; otherwise, it would have been sufficient for s'_j to serve the scheduled request pairs and $\{r_{h,1}, r_{h,2}\}$. We charge $\{r_{h,1}, r_{h,2}\}$ to $\{r_{c,1}, r_{c,2}\}$. In this way, every request pair in $\bar{R}^*_j$ is charged to a request pair in R'_j .

Then we bound the number of request pairs that can be charged to a single request pair $\{r_{c,1}, r_{c,2}\}$ in R'_j . By the above charging scheme and recall Observation 1, every request that is served by s_j^* and charges to $r_{c,1}$ (resp. $r_{c,2}$) has a start time in $(t_{c,1} - 2t, t_{c,1} + 2t)$ (resp. $(t_{c,2} - 2t, t_{c,2} + 2t)$) and pick-up location $p_{c,1}$ (resp. $p_{c,2}$), or has a start time in $(t_{c,1} - t, t_{c,1} + t)$ (resp. $(t_{c,2} - t, t_{c,2} + t)$) and pick-up location $\dot{p}_{c,1}$ (resp. $\dot{p}_{c,2}$). As all requests have travel time at least t , the start times of consecutive requests (with the same pick-up location) accepted by s_j^* differ by at least $2t$. Therefore, $\bar{R}_j^*$ contain at most $(2 + 1) \times 2 = 6$ request pairs that charges to $\{r_{c,1}, r_{c,2}\}$. Hence $\{r_{c,1}, r_{c,2}\}$ is charged by at most 6 request pairs from $\bar{R}_j^*$. This establishes the claim, with $\alpha = 6$. $\square$

3.2 Balanced greedy algorithm

In this section, we propose a balanced greedy algorithm (BGA) for ONLINETRANSFERSFORCOMMUTING with slotted pick-up times, shown in Algorithm 5. Recall that the pick-up time $t_{i,j}$ is an integer multiple of the travel time t , i.e., $t_{ij} = v \cdot t$ for some $v \in \mathbb{N}$. $R_{i,j}$ is the set of request pairs whose pick-up location is $p(i, j)$ and pick-up time is $t(i, j)$. If a car serves a request $r_{i,j}$, we say that this car stops at location $\dot{p}_{i,j}$ at time $t_{i,j} + t$.

Algorithm 5 Balanced greedy algorithm (BGA)

- 1: *Input:* k cars, request pairs arrive over time.
 - 2: *Step 1:* When a request pair $\{r_{i,1}, r_{i,2}\}$ arrives
 - 3: If $|R_{i,1}| < k/2$ and $|R_{i,2}| < k/2$,
 - 4: accept $\{r_{i,1}, r_{i,2}\}$;
 - 5: Otherwise, reject it.
 - 6: *Step 2:* Assign request $r_{i,j}$ at time $t_{i,j} - t$.
 - 7: If $|S_{i,j}| \geq 1$,
 - 8: assign it to a car in $S_{i,j}$;
 - 9: Otherwise, assign it to a car who is not assigned to any requests starting at or after $t_{i,j} - t$.
- * $S_{i,j}$ is the set of cars who stop at $p_{i,j}$ at time $t_{i,j}$ and has not be assigned to any requests starting at $t_{i,j}$.
-

Lemma 3 *In Algorithm 5, all requests accepted in Step 1 could be assigned to a car in Step 2.*

Proof Observe that all requests start no earlier than t since $b_i \geq t$. We claim that before assigning requests with pick-up time it ($i \in \mathbb{N}$), at most $k/2$ cars stop at location 0 at time it and at most $k/2$ cars stop at location 1 at time it . If this claim holds, we could assign each accepted request with pick-up time it to a car by Step 2 in Algorithm 5 since the number of accepted requests with the same pick-up time and pick-up location is not greater than $k/2$, proving the lemma.

In the following, we will prove this claim by induction over the times it ($i \in \mathbb{N}$).

Base Case Initially, all cars stay at location 0. Consider base case $i = 1$ and $i = 2$.

$i = 1$, consider the case of assigning requests with pick-up time t . Based on the problem setting that booking interval is greater than t , then no requests starts before t , and thus, no car stops at location 0 at time t and no car stops at location 1 at time t . $i = 2$, consider the case of assigning requests with pick-up time $2t$. According to *Step 1* in Algorithm 5, we know that BGA accepts no more than $k/2$ requests with pick-up time t and pick-up location 0 (resp. 1), thus, we know that at most $k/2$ cars stop at location 0 (resp. 1) at time $2t$.

Induction step We assume that at most $k/2$ cars stop at location 0 at time jt and at most $k/2$ cars stop at location 1 at time jt hold for all instances with $j \leq i$ and show that then at most $k/2$ cars stop at location 0 at time $(i + 1)t$ and at most $k/2$ cars stop at location 1 at time $(i + 1)t$ also hold for all instances.

Consider an instance of $OTFC - k, R$, where the pick-up time of the last request in R is it . Let $R'(i, 0)$ be the set of requests in R' with pick-up time it and pick-up location 0, $R'(i, 1)$ be the set of requests in R' with pick-up time it and pick-up location 1. Since BGA accepts request pairs $\{r_{i,1}, r_{i,2}\}$ if and only if $|R'_{i,1}| < k/2$ and $|R'_{i,2}| < k/2$ (see Step 1, GA at most accept $k/2$ requests with the same pick-up location and pick-up time), the number of accepted requests with pick-up time it and pick-up location 0 (resp. 1) is not greater than $k/2$. In other words, $|R'(i, 0)| \leq k/2$ and $|R'(i, 1)| \leq k/2$.

Suppose there are k_0 cars that stop at location 0 at time it , k_1 cars that stop at location 1 at time it , and the set of the rest cars is denoted by S_{res} . By the inductive hypothesis, $k_0 \leq k/2$, $k_1 \leq k/2$ and $|S_{res}| = (k/2 - k_0) + (k/2 - k_1)$. Based on *Step 2* in Algorithm 5, $\min\{k_0, |R'(i, 0)|\}$ cars serve requests in $R'(i, 0)$ and $\min\{k_1, |R'(i, 1)|\}$ cars serve requests in $R'(i, 1)$. If $k_0 < |R'(i, 0)|$ (resp. $k_1 < |R'(i, 1)|$), BGA assigns rest $|R'(i, 0)| - \min\{k_0, |R'(i, 0)|\}$ (resp. $|R'(i, 1)| - \min\{k_1, |R'(i, 1)|\}$) requests to the cars in S_{res} . Since the cars in S_{res} can reach location 0 or 1 at time it , all requests in $R'(i, 1)$ and $R'(i, 1)$ could be served by the assigned cars. After assigning all requests with pick-up time it , $|R'(i, 0)| \leq k/2$ cars stop at location 0 at time $(i + 1)t$ and $|R'(i, 1)| \leq k/2$ cars stop at location 1 at time $(i + 1)t$. $\square$

Theorem 8 For $b_u \geq b_l \geq t$, BGA is 4-competitive for $OTFC - k$ ($k = 2i, i \in \mathbb{N}$) with variable booking times and slotted pick-up times.

Proof For the purpose of the analysis, we partition the set of accepted request pairs R' into a number of request sets R'_i with $1 \leq i \leq 2n$, where $n = \frac{t_{R,1} - t_{1,1}}{t}$. R'_{2i-1} contains the requests with pick-up time $t_{1,1} + it$ and pick-up location 0, and R'_{2i} contains the requests with pick-up time $t_{1,1} + it$ and pick-up location 1. The requests in set $\bigcup R'_i$ are indexed in order of non-decreasing pick-up times, and for the request with the same pick-up times, they are indexed by the time they are released. Note that each request $r_{g,h}$ has a unique mapping, defined by $R'_{f(r_{g,h})}$ ($1 \leq f(r_{g,h}) \leq 2n$), which contains all accepted requests with pick-up time $t_{g,h}$ and pick-up location $p_{g,h}$. Observe that $|R'| = \frac{1}{2} \sum_{i=1}^{2n} |R'_i|$.

Let $\bar{R}^*$ be the set of request pairs accepted by OPT , but did not accepted by BGA. By *Step 1* in Algorithm 5, we know that BGA will reject a request pair $\{r_{i,1}, r_{i,2}\}$ if and only if BGA has accepted $k/2$ request pairs which contains a request with pick-up location $t_{i,1}$ and pick-up time $p_{i,1}$, or BGA has accepted $k/2$ request pairs which

contains a request with pick-up location $t_{i,2}$ and pick-up time $p_{i,2}$. That means for any request pair $\{r_{i,1}, r_{i,2}\} \in \bar{R}^*$, either $|R'_{f(r_{i,1})}| = k/2$, or $|R'_{f(r_{i,2})}| = k/2$, or both. Let R'_a be the union of all such request set $R'_{f(r_{i,1})}$ ($= k/2$) and $R'_{f(r_{i,2})}$ ($= k/2$). For $R'_{f(r_{i,1})} \in R'_a$, let $R^*_{f(r_{i,1})}$ be the set of request pairs accepted by OPT in which one of the request starts at $t_{i,1}$ at $p_{i,1}$. Similarly, for $R'_{f(r_{i,2})} \in R'_a$, let $R^*_{f(r_{i,2})}$ be the set of request pairs accepted by OPT in which one of the request starts at $t_{i,2}$ at $p_{i,2}$. Let R^*_a be the union of all such $R^*_{f(r_{i,1})}$ and $R^*_{f(r_{i,2})}$. Note that $\bar{R}^* \subset R^*_a$.

We charge requests in $R^*_{f(r_{i,1})}$ by the requests in $R'_{f(r_{i,1})}$. Since $|R^*_{f(r_{i,1})}| \leq k$ where each item in $R^*(i, 1)$ is a request pair and $|R'_{f(r_{i,1})}| = k/2$, the number of requests in $R^*_{f(r_{i,1})}$ is not greater than four times of the requests in $R'_{f(r_{i,1})}$. The other case $R^*_{f(r_{i,2})}$ is symmetric, i.e., the number of requests in $R^*_{f(r_{i,2})}$ is not greater than four times of the requests in $R'_{f(r_{i,2})}$. Summing up, the number of requests in R'_a is not greater than four times of the requests in R^*_a .

Since all request pairs in $R^* \setminus R^*_a$ are accepted by BGA and these requests are not contained in R'_a , we know that the requests in $R^* \setminus R^*_a$ is a subset of the requests in $\bigcup R' \setminus R'_a$. Therefore the requests in $R^* \setminus R^*_a$ is not greater than the requests in $\bigcup R'_i \setminus R'_a$.

The above analysis establishes the theorem, with $|R^*|/|R'| \leq 4$. $\square$

Remark When $k = 2i + 1$ ($i \in \mathbb{N}$), based on *Step 1* in Algorithm 5 and the analysis in Theorem 8, we know that for any request pair $\{r_{i,1}, r_{i,2}\} \in \bar{R}^*$, either $|R'_{f(r_{i,1})}| = (k-1)/2$, or $|R'_{f(r_{i,2})}| = (k-1)/2$, or both. Since $|R^*_{f(r_{i,1})}| \leq k$ (resp. $|R^*_{f(r_{i,2})}| \leq k$) where each item in $R^*(i, 1)$ (resp. $R^*(i, 2)$) is a request pair, the number of requests in R'_a is not greater than $\frac{2k}{(k-1)/2}$ times of the requests in R^*_a . Recall that the requests in $R^* \setminus R^*_a$ is not greater than the requests in $\bigcup R'_i \setminus R'_a$, we know that $|R^*|/|R'| \leq 4 + \frac{4}{k-1}$.

4 Conclusion

In this work, we studied the $ONLINE TRANSFERS FOR COMMUTING$ ($OTFC - k$) problem that is motivated by applications of car-sharing platform. In our model, each customer submits a request pair between the two locations. For a single car, we analyzed the effects that different constraints on the booking time of request pairs on the competitive ratio that can be achieved. We proved that no deterministic online algorithm for $OTFC - 1$ with fixed booking times $0 \leq a < t$ can achieve a constant competitive ratio. For $OTFC - 1$ with fixed booking times $a \geq t$, we established a bound of 5 on the competitive ratio; and for $OTFC - 1$ with variable booking times, we established a bound of 6 on the competitive ratio. For $OTFC - k$ ($k \geq 1$) with fixed booking times with $t \leq a$ or with the variable booking times with $b_u > b_l \geq t$, we gave the matching lower and upper bounds on the competitive ratio. The upper bounds are all achieved by the same Greedy Algorithm (GA). For $OTFC - k$ ($k > 2$) problem with variable booking times ($b_u > b_l \geq t$ and $b_u > b_l + t$) and slotted pick-up times, we established a bound of 4 on the competitive ratio. Furthermore, we presented a 4-competitive algorithm, balanced greedy algorithm (BGA).

Open problems We studied the car-sharing problem with two locations, therefore, a natural direction is to explore it with many locations in different networks, such as a star network.

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement Number 754462, this research is partially supported by the National Natural Science Foundation of China (Grant Nos. 71601152, 71701048).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bar-Yehuda R, Halldórsson MM, Naor J, Shachnai H, Shapira I (2006) Scheduling split intervals. *SIAM J Comput* 36:1–15
- Bei X, Zhang S (2018) Algorithms for trip-vehicle assignment in ride-sharing. In: Thirty-second AAAI conference on artificial intelligence, pp 3–9
- Böhmová K, Disser Y, Mihalák M, Šrámek R (2016) Scheduling transfers of resources over time: towards car-sharing with flexible drop-offs. In: *LATIN 2016: theoretical informatics*. Springer, pp 220–234
- Borodin A, El-Yaniv R (2005) *Online computation and competitive analysis*. Cambridge University Press, Cambridge
- Dutta C, Sholley C (2018) Online matching in a ride-sharing platform. *arXiv preprint arXiv:1806.10327*
- Golumbic MC (2004) *Algorithmic graph theory and perfect graphs*. Elsevier, Amsterdam
- Gyárfás A, West D (1995) Multitrack interval graphs
- Lipton R (1994) Online interval scheduling. In: *Proceedings of the fifth annual ACM-SIAM symposium on discrete algorithms*. Society for Industrial and Applied Mathematics, pp 302–311
- Liu H, Luo K, Xu Y, Zhang H (2019) Car-sharing problem: online scheduling with flexible advance bookings. In: *international conference on combinatorial optimization and applications*. Springer, pp 340–351
- Luo K, Erlebach T, Xu Y (2018a) Car-sharing between two locations: online scheduling with flexible advance bookings. In: *International computing and combinatorics conference*. Springer, pp 242–254
- Luo K, Erlebach T, Xu Y (2018b) Car-sharing between two locations: online scheduling with two servers. In: *43rd international symposium on mathematical foundations of computer science (MFCS)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, pp 50:1–50:14
- Luo K, Erlebach T, Xu Y (2018c) Online scheduling of car-sharing requests between two locations with many cars and flexible advance bookings. In: *29th international symposium on algorithms and computation (ISAAC)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, pp 64:1–64:13
- Luo K, Erlebach T, Xu Y (2019a) Car-sharing on a star network: on-line scheduling with k servers. In: *36th international symposium on theoretical aspects of computer science (STACS)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, pp 51:1–51:14
- Luo K, Xu Y, Liu H (2019b) Online scheduling of car-sharing request pairs between two locations with advance bookings. In: *2019 international conference on industrial engineering and systems management (IESM)*. IEEE, pp 1–6

- Nayyar K, Raghvendra S (2017) An input sensitive online algorithm for the metric bipartite matching problem. In: 2017 IEEE 58th annual symposium on foundations of computer science (FOCS). IEEE, pp 505–515
- Raghvendra S (2018) Optimal analysis of an online algorithm for the bipartite matching problem on a line. arXiv preprint [arXiv:1803.07206](https://arxiv.org/abs/1803.07206)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.