Car-Sharing between Two Locations: Online Scheduling with Flexible Advance Bookings [⋆]

Kelin Luo[a,*], Thomas Erlebach[b], Yinfeng Xu[c,d]

[a]*Department of Mathematics and Computer Science, Eindhoven University of Technology, the Netherlands*
[b]*School of Informatics, University of Leicester, United Kingdom*
[c]*School of Management, Xi'an Jiaotong University, China*
[d]*The State Key Lab for Manufacturing Systems Engineering, Xi'an, China*

Abstract

We study an online scheduling problem that is motivated by applications such as car-sharing. Users submit ride requests, and the scheduler aims to accept requests of maximum total profit using a single server (car). Each ride request specifies the pick-up time and the pick-up location (among two locations, with the other location being the destination). The scheduler has to decide whether or not to accept a request immediately at the time when the request is submitted (booking time). We consider two variants of the problem with respect to constraints on the booking time: In the fixed booking time variant, a request must be submitted a fixed amount of time before the pick-up time. In the variable booking time variant, a request can be submitted at any time during a certain time interval that precedes the pick-up time. We present lower bounds on the competitive ratio of deterministic and randomized algorithms for both variants and propose a greedy algorithm that achieves the best possible competitive ratio among deterministic algorithms.

Keywords: scheduling, competitive analysis, booking horizon

[⋆]An extended abstract of this paper has appeared in the *Proceedings of the 24th International Conference on Computing and Combinatorics* (COCOON 2018), LNCS 10976, Springer, 2018, pp. 242–254.

[*]Corresponding author

Email addresses: `luo.kelin@outlook.com` (Kelin Luo), `te17@leicester.ac.uk` (Thomas Erlebach), `yfxu@xjtu.edu.cn` (Yinfeng Xu)

1. Introduction

In a car-sharing system, a company offers cars to customers for a period of time. Customers can pick up a car in one location, drive it to another location, and return it there. Car booking requests arrive online, and the goal is to maximize the profit obtained from satisfied requests. We consider a setting where all driving routes go between two fixed locations, but can be in either direction. For example, the two locations could be a residential area and a nearby shopping mall or central business district. Other applications that provide motivation for the problems we study include taxi dispatching and boat rental for river crossings.

In real life, customer requests for car bookings usually arrive over time, and the decision about each request must be made immediately, without knowledge of future requests. This gives rise to an online problem that bears some resemblance to interval scheduling, but in which additionally the pick-up and drop-off locations play an important role: The server that serves a request must be at the pick-up location at the start time of the request and will be located at the drop-off location at the end time of the request. A server can serve two consecutive requests only if the drop-off location of the first request is the same as the pick-up location of the second request, or if there is enough time to travel between the two locations otherwise. (We allow 'empty movements' that allow a server to be moved from one location to another while not serving a request. Such empty movements could be implemented by having company staff drive a car from one location to another, or in the future by self-driving cars.)

An important aspect of the problem is the relation between the *booking time*, i.e., the time when the request is submitted, and the *start time* of the request, i.e., the time when the customer picks up the car at the pick-up location. Constraints on the booking time (also called the *reservation window* in the context of advance reservation systems) can affect the performance of a system [1]. There are generally two types of bookings, current and advance. Current bookings are requests that are released and must be served immediately. Advance bookings are requests that are released before the start time. In this paper we consider advance bookings. More specifically, we study two variants of advance bookings: In the *fixed booking time* variant, the amount of time between the booking time of a request and its start time is a fixed value, independent of the request. In the *variable booking time* variant, the booking time of a request must lie in a certain time interval (called the *booking horizon*) before the start time of the request.

We assume that every request is associated with a profit that is obtained if the request is accepted. When a server moves from one location to another while not serving a request, a certain cost is incurred. The goal is to maximize the total profit, which is the sum of the profits of the accepted requests minus the costs incurred for moving servers while not serving a request. In this paper, we focus on the special case of a single server.

1.1. Related Work

The car sharing problem considered in this paper belongs to the class of dynamic pick-up and delivery problems surveyed by Berbeglia et al. [2]. The problem that is closest to our setting is the online dial-a-ride problem (OLDARP) that has been studied widely. In OLDARP, transportation requests between locations in a metric space arrive over time, but typically it is assumed that requests want to be served 'as soon as possible' rather than at a specific time as in our problem. Known results for OLDARP include online algorithms for minimizing the makespan [3, 4] or the maximum flow time [5]. Grötschel et al. [6] use simulations to compare several online algorithms for OLDARP, considering the objectives of minimizing the average flow time and minimizing the maximum flow time. One of the algorithms, called IG-GREEDY, adds a new request to the current schedule immediately if it can be inserted into the current schedule without increasing the cost, and ignores the new request for now and schedules it later otherwise. In our problem, we do not need to serve all requests, but we consider a similar greedy algorithm: Our algorithm accepts and serves a request if it can be served together with the previously accepted requests and if serving this request increases the profit.

Work on versions of OLDARP where not all requests can be served includes competitive algorithms for requests with deadlines where each request must be served before its deadline or rejected [7], and for settings with a given time limit where the goal is to maximize the revenue from requests served before the time limit [8]. In contrast to existing work on OLDARP, in this paper we consider requests that need to be served at a specific time that is specified by the request when it is released.

Offline versions of car-sharing problems are studied by Böhmová et al. [9]. They show that if all customer requests for car bookings are known in advance, the problem of maximizing the number of accepted requests can be solved in polynomial time using a minimum-cost network flow algorithm [10].

In the instance of the minimum-cost network flow problem that they construct, each arc connecting a pick-up location/time and a drop-off location/time has cost -1 and all others have cost 0, as their objective is to maximize the number of accepted requests. This approach can also be adapted to solve the offline version of the problem considered in this paper optimally in polynomial time: We construct an instance of the minimum-cost network flow problem as follows. The network contains a node for the pick-up location/time and the drop-off location/time of each request, a source node s, a target node t, and an auxiliary node h. There are four types of arcs: For each request, there is an arc with capacity 1 and cost $-r$ (the profit of accepting one request) from its pick-up location/time to its drop-off location/time; for every two requests that can be accepted consecutively by one car, there is an arc connecting the drop-off location/time of the first request and the pick-up location/time of the second request with capacity 1 and cost 0 if the two locations are the same or cost c (the cost of an empty movement from one location to another) otherwise; for each request, there is an arc from h to its pick-up location/time with capacity 1 and cost 0 if the pick-up location is the original location of the car or cost c otherwise, and there is an arc from its drop-off location/time to t with capacity 1 and cost 0; finally, there is an arc from s to h with capacity 1 (for one car, or capacity k if there are k cars) and cost 0. A minimum-cost maximum s-t-flow gives an optimal solution to our car-sharing problem, where the objective value of the flow is the negative of the profit of the solution to the car-sharing problem. Furthermore, Böhmová et al. [9] consider the problem variant with two locations where each customer requests two rides (in opposite directions) and the scheduler must accept either both or neither of the two. They prove that this variant is NP-hard and APX-hard. In contrast to their work, we consider the online version of the problem.

Online TSP problems (see, e.g., [11, 12]) are also related to our work, but less directly because in these problems requests are served at a specific location rather than involving a pick-up and a delivery, and typically the requests do not need to be served at a fixed time.

1.2. Problem Description and Preliminaries

We consider a setting with only two locations (denoted by 0 and 1) and a single server. The travel time from 0 to 1 is the same as the travel time from 1 to 0 and is denoted by t. Let R denote a sequence of requests that are released over time. The i-th request is denoted by $r_i = (\tilde{t}_{r_i}, t_{r_i}, p_{r_i})$ and is specified by the *booking time* or *release time* $\tilde{t}_{r_i}$, the *start time* t_{r_i}, and the pick-up location $p_{r_i} \in \{0, 1\}$. Note that the drop-off location of r_i is the location that is different from the pick-up location, i.e., $1 - p_{r_i}$. Requests with the same release time arrive one by one in arbitrary order, and each request must be processed by the algorithm before the next request arrives. If r_i is accepted, the server must pick up the customer at p_{r_i} at time t_{r_i} and drop off the customer at location $\dot{p}_{r_i} = 1 - p_{r_i}$ at time $\dot{t}_{r_i} = t_{r_i} + t$, the *end time* of the request. We say that the request r_i *starts* at time t_{r_i}. For an interval $[b, d)$, we say that r_i starts in the interval if $t_{r_i} \in [b, d)$.

The server can only serve one request at a time. Serving a request yields profit $r > 0$. The server is initially located at location 0. If the pick-up location p_{r_i} of a request r_i is different from the current location of the server and if at least t time units remain before the start time of r_i, the server can move from its current location to p_{r_i}. We refer to such moves (which do not serve a request) as *empty* moves. An empty move takes time t and incurs a cost of c, $r \geq c \geq 0$, and we say that r_i is accepted *with cost* in this case. If the server is already located at p_{r_i}, we say that r_i is accepted *without cost*. As motivation for the cost model, we can imagine that the customer pays a fee of f, where $f > c$, for being served, while the cost (in terms of fuel etc.) of moving the car from one location to the other is c. Then an empty move incurs a cost of c, while serving a request yields a profit of $r = f - c > 0$. We can assume that $f \geq 2c$, implying that $r \geq c$.

If two requests are such that they cannot both be served by one server, we say that the requests are *in conflict*. We forbid 'unprompted' moves for the offline adversary (or, equivalently, for the optimal offline solution to which we compare the solution produced by an online algorithm), i.e., the offline adversary is allowed to make an empty move to the other location only if it does so in order to serve a request that was accepted before the current time and whose pick-up location is the other location. The offline adversary is allowed to make an empty move to the other location only if its currently next unserved request (i.e., the request that has already been accepted but not yet served and has earliest pick-up time among all such requests) has that location as the pick-up location. Our greedy algorithm does not use unprompted moves. All our lower bounds also apply to algorithms that may use unprompted moves.

We denote the requests accepted by an algorithm by R', and the i-th request in R', in order of request start times, is denoted by r'_i. We say that request r'_i is accepted *without cost* if $i = 1$ and $p_{r'_i} = 0$ or if $i > 1$ and

$p_{r'_i} = \dot{p}_{r'_{i-1}}$. Otherwise, r'_i is accepted *with cost*. We denote the profit of serving the requests in R' by $P_{R'}$. If R'_c denotes the subset of R' consisting of the requests that are accepted with cost, we have $P_{R'} = r \cdot |R'| - c \cdot |R'_c|$.

The goal is to accept a set of requests R' that maximizes the profit $P_{R'}$. The problem for one server and two locations for the fixed booking time variant in which $t_{r_i} - \tilde{t}_{r_i} = a$ for all requests r_i, where $a \geq 0$ is a constant, is called the *1S2L-F problem*. For the variable booking time variant, the booking time $\tilde{t}_{r_i}$ of any request r_i must satisfy $t_{r_i} - b_u \leq \tilde{t}_{r_i} \leq t_{r_i} - b_l$, where b_l and b_u are constants, with $b_l \leq b_u$, that specify the minimum and maximum length, respectively, of the time interval between booking time and start time. The problem for one server and two locations for the variable booking time variant is called the *1S2L-V problem*. If $b_l = b_u$, the 1S2L-V problem turns into the 1S2L-F problem.

We briefly discuss the case $r = c$, where a request accepted with cost yields profit $r - c = 0$. For 1S2L-F, an algorithm will never make an empty move to accept a request in this case, so the case $r = c$ is equivalent to forbidding empty moves. For 1S2L-V, however, the following is possible if $r = c$: An algorithm may first accept a request r_i from 1 to 0 with cost (i.e., making an empty move), gaining a profit of zero. Afterwards, a request from 0 to 1 may arrive that has an earlier start time than r_i and can be served before r_i, replacing the empty move from 0 to 1. In this way, the algorithm can gain profit $2r$ from the two requests. In fact, the issue that accepting requests with empty moves can turn out to be useful later on if suitable other requests arrive is one of the most interesting aspects of the problem variant with variable booking time for $r = c$, see Theorems 6 and 10. In a real-life scenario, the case $r = c$ arises if the fee f paid by the customer is exactly twice the cost c of moving the car from one location to the other.

The performance of an algorithm for 1S2L-F or 1S2L-V is measured using competitive analysis (see [13, 14]). For any request sequence R, let P_{R^A} denote the objective value produced by an online algorithm A, and P_{R^*} that obtained by an optimal scheduler OPT that has full information about the request sequence in advance. This paper presents a simple deterministic algorithm with optimal competitive ratio for the above two problems. Recall that we require that OPT does not make unprompted moves, i.e., OPT is allowed to make an empty move starting at time t_0 only if there is an accepted request r_i with $\tilde{t}_{r_i} \leq t_0$ and $t_{r_i} \geq t_0 + t$ whose pick-up location is the other location. Without this restriction on OPT, it would not be possible for deterministic algorithms to achieve finite competitive ratio in cases where a request can be booked less than t units of time before its starting time. (A randomized algorithm can be competitive even if OPT is allowed to make unprompted moves.)

The competitive ratio of A is defined as $\rho_A = \sup_R \frac{P_{R^*}}{P_{R^A}}$. We say that A is ρ-competitive if $P_{R^*} \leq \rho \cdot P_{R_A}$ for all request sequences R. (For a randomized algorithm A, P_{R_A} is replaced by the expected profit $\mathbf{E}[P_{R_A}]$ in this definition.) If there is a ρ-competitive algorithm, ρ is an *upper bound* (UB) on the best possible competitive ratio that can be achieved for the problem. Let ON be the set of all online algorithms for a problem. A value β is a *lower bound* (LB) on the best possible competitive ratio if it can be proven that $\rho_A \geq \beta$ for all A in ON. For ON we may consider all deterministic algorithms or we may also allow randomized algorithms. Lower bounds are proven using adversarial arguments and hold for all algorithms. Upper bounds are proven by analyzing a specific algorithm and showing that the algorithm is ρ-competitive. The aim is to prove matching upper and lower bounds. In this case, if there is a lower bound β and an algorithm A with $\rho_A = \beta$, we say that A is optimal (among all online algorithms).

1.3. Paper Outline

In Section 2, we study the 1S2L-F problem. We give lower bounds and propose a Greedy Algorithm (GA) that accepts each request if it can be served and is profitable. We show that the algorithm achieves the best possible competitive ratio among all deterministic online algorithms. In Section 3, we study the 1S2L-V problem. Although variable booking times provide much greater flexibility to customers, we show that GA is still optimal among deterministic algorithms. An overview of our results is shown in Table 1. The rows of the table cover the whole range of possible parameter values for the booking time constraint, i.e., for how the parameter a relates to t for 1S2L-F and for how the parameter b_u relates to t for 1S2L-V. For each setting, we consider the case $0 \leq c < r$ (where a request that is accepted with cost is also profitable) and the case $c = r$ (where a request accepted with cost has zero profit but could, in the 1S2L-V setting, potentially become profitable if serving a request with earlier start time that arrives later can replace the empty movement). For each case, our lower bound on the best competitive ratio of any deterministic algorithm matches the upper bound we obtain with algorithm GA, indicated by the 'LB=UB (det)' columns. Furthermore, we give lower bounds for randomized algorithms, indicated by the 'LB (rand)' columns. The table entries also indicate the theorems where the corresponding bounds are proved. A lower bound of 1 holds trivially for each problem variant.

Table 1: Lower and upper bounds for the car sharing problem

		$0 \leq c < r$		$c = r$	
Problem	Parameter	LB=UB (det)	LB (rand)	LB=UB (det)	LB (rand)
1S2L-F	$0 \leq a < t$	1 (Th. 2)	1	1 (Th. 2)	1
1S2L-F	$t \leq a$	$\frac{2r}{r-c}$ (Th. 1,3)	$\frac{3r+c}{2r}$ (Th. 1)	1 (Th. 2)	1
1S2L-V	$0 < b_u < t$	3 (Th. 5, 7)	5/3 (Th. 5)	3 (Th. 6,10)	5/3 (Th. 6)
1S2L-V	$b_u = t$	$\max\{\frac{2r}{r-c}, 3\}$ (Th. 2,5,8)	5/3 (Th. 5)	3 (Th. 6,10)	5/3 (Th. 6)
1S2L-V	$t < b_u$	$\frac{3r-c}{r-c}$ (Th. 4,9)	$\frac{5r-c}{3r-c}$ (Th. 4)	$1 + 2\lceil\frac{b_u - b_l}{2t}\rceil$ (Th. 6,10)	5/3 (Th. 6)

2. Car Sharing with Fixed Booking Times

In this section, we study the 1S2L-F problem. First, we present a lower bound. We use ALG to denote any online algorithm and OPT to denote an optimal scheduler. We refer to the server of ALG and OPT as s' and s^*, respectively. The set of requests accepted by ALG is referred to as R', and the set of requests accepted by OPT as R^*.

Theorem 1. *For $0 \leq c < r$ and $a \geq t$, no deterministic online algorithm for 1S2L-F can achieve a competitive ratio smaller than $\frac{2r}{r-c}$. Furthermore, no randomized online algorithm can achieve a competitive ratio smaller than $\frac{3r+c}{2r}$.*

Proof. We first present the lower bound for deterministic algorithms. Initially, the adversary releases a request $r_1 = (0, a, 1)$. We distinguish two cases.

Case 1: ALG accepts r_1 (with cost). The adversary releases requests $r_2 = (\varepsilon, a + \varepsilon, 0)$ and $r_3 = (\varepsilon + t, a + \varepsilon + t, 1)$, where $0 < \varepsilon < t$. OPT accepts r_2 and r_3 without cost, but ALG cannot accept either of these requests as they are in conflict with r_1 (see Fig. 1 for an illustration). We have $P_{R^*} = 2r$ and $P_{R'} = r - c$, and hence $\frac{P_{R^*}}{P_{R'}} = \frac{2r}{r-c}$.

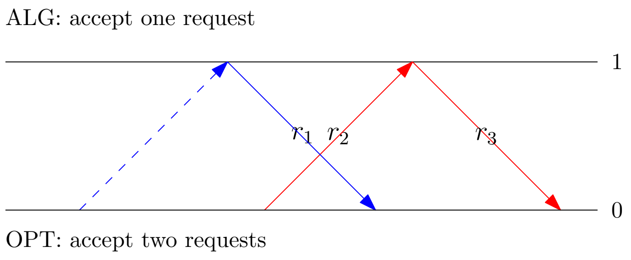


Figure 1: Illustration of R^* and R' in Case 1

Case 2: ALG does not accept request r_1. In this case, OPT accepts r_1 and we have $P_{R^*} = r - c$ and $P_{R'} = 0$, and hence $\frac{P_{R^*}}{P_{R'}} = \infty$.

To prove a lower bound for randomized algorithms, we apply Yao's principle [13]: Giving a probability distribution $y(j)$ over request sequences σ_j and showing that

$$\frac{\mathbf{E}_{y(j)}[OPT(\sigma_j)]}{\mathbf{E}_{y(j)}[A(\sigma_j)]} \geq \beta$$

holds for all deterministic algorithms A implies a lower bound of β on the competitive ratio of any randomized online algorithm.

Consider the following probability distribution $y(j)$ over request sequences: We present request sequence $\sigma_1 = \{r_1\}$ with probability $y(1) = p = \frac{r+c}{2r}$ and request sequence $\sigma_2 = \{r_1, r_2, r_3\}$ with probability $y(2) = 1 - p$, where r_1, r_2, r_3 are defined as above. When r_1 is released, every deterministic algorithm either accepts it or rejects it. Deterministic algorithms that accept r_1 (denoted by ALG_1) gain profit $r - c$ on both sequences. Deterministic algorithms that reject r_1 (denoted by ALG_2) gain profit 0 on σ_1 and profit at most $2r$ on σ_2, thus an expected profit of at most $p \cdot 0 + (1-p)2r = r - c$. The profit gained by OPT is $r - c$ on σ_1 and $2r$ on σ_2, thus an expected

Table 2: Profits for σ_1 and σ_2 in the proof of Theorem 1

	σ_1	σ_2
ALG_1	$r-c$	$r-c$
ALG_2	0	$2r$
OPT	$r-c$	$2r$

profit of $p(r-c) + (1-p)2r = \frac{(r-c)(3r+c)}{2r}$. See Table 2 for an overview. We have

$$\frac{\mathbf{E}_{y(j)}[OPT(\sigma_j)]}{\mathbf{E}_{y(j)}[A(\sigma_j)]} \geq \frac{\frac{(r-c)(3r+c)}{2r}}{r-c} = \frac{3r+c}{2r},$$

and by Yao's principle it follows that no randomized online algorithm can achieve competitive ratio smaller than $\frac{3r+c}{2r}$. □

Algorithm 1 Greedy Algorithm (GA)

```
Input: one server, requests arrive over time.
Step: When request $r_i$ arrives, accept $r_i$ if $r_i$ is acceptable and $P_{R_i^{GA}\cup\{r_i\}} - P_{R_i^{GA}} > 0$;
```

Note 1: R_i^{GA} is the set of requests accepted by GA before r_i is released.
Note 2: r_i is acceptable if and only if $\forall r'_j \in R_i^{GA}, |t_{r_i} - t_{r'_j}| \geq 2t$ if $p_{r_i} = p_{r'_j}$, and $|t_{r_i} - t_{r'_j}| \geq t$ if $p_{r_i} \neq p_{r'_j}$, and $t_{r_i} - \tilde{t}_{r_i} \geq t$ if s' is at location $p_{s'} \in \{0,1\}$ at time $\tilde{t}_{r_i}$ and $p_{r_i} = 1 - p_{s'}$.

We propose a Greedy Algorithm (GA) for the 1S2L-F problem, shown in Algorithm 1. The algorithm is based on the natural idea of accepting each request if it can be served and is profitable. For an arbitrary request sequence $R = \{r_1, r_2, r_3, \ldots, r_n\}$, note that we have $t_{r_i} \leq t_{r_{i+1}}$ for $1 \leq i < n$ because $t_{r_i} - \tilde{t}_{r_i} = a$ is fixed. Denote the requests accepted by OPT by $R^* = \{r_1^*, r_2^*, \ldots, r_{k^*}^*\}$ and the requests accepted by GA by $R' = \{r'_1, r'_2, \ldots, r'_k\}$ indexed in order of non-decreasing start times.

We compare the profit gained by OPT and GA as follows. In Theorem 2, we consider the setting where both OPT and GA only accept requests without cost, and in the proof we show that the schedule of OPT can be transformed into GA's schedule without reducing the profit. This is because, at any point in the schedule, the next request of the remaining schedule by OPT can always be replaced by the next request of GA's schedule, which is the earliest acceptable request at that point. This establishes that GA is 1-competitive. Theorem 3 considers the remaining setting, where $c < r$ and $a \geq t$. Here, we partition the schedule into periods in such a way that GA accepts one request that starts in each period. We then analyze each period separately and check the maximum profit that could be gained by OPT in the period. As the requests accepted by OPT cannot start before the request accepted by GA because of the definition of periods, we can show that OPT can accept at most two requests in the period. Thus, we have that GA achieves profit r or $r - c$ in the period, while OPT gains at most $2r$.

Theorem 2. *Algorithm GA is* 1*-competitive for 1S2L-F if* $c = r$*, or if* $0 \leq c < r$ *and* $0 \leq a < t$*.*

Proof. If $0 \leq c < r$ and $0 \leq a < t$, GA and OPT only accept requests without cost because the release time of a request is too late for the server to be able to serve it with cost (recall that we forbid unprompted moves by OPT). Observe that this means that both GA and OPT accept requests with alternating pick-up location, starting with a request with pick-up location 0.

We claim that R^* can be transformed into R' without reducing its profit, thus showing that $P_{R^*} = P_{R'}$. As GA accepts the first request r_j with $p_{r_j} = 0$, it is clear that $t_{r'_1} \leq t_{r^*_1}$. If $r'_1 \neq r^*_1$, we can replace r^*_1 by r'_1 in R^*, and R^* is still a valid solution with the same profit. Now assume that R' and R^* are identical with respect to the first i requests, and that s' and s^* are at location $p \in \{0, 1\}$ at time $\dot{t}_{r'_i}$. If there is a request r^*_{i+1}, there must also be a request r'_{i+1} with $t_{r'_{i+1}} \leq t_{r^*_{i+1}}$, as GA could accept r^*_{i+1}. We can replace r^*_{i+1} by r'_{i+1} in R^*. The claim thus follows by induction.

If $c = r$, accepting a request with cost yields profit $r - c = 0$. Without loss of generality, we can therefore assume that both GA and OPT only accept requests without cost. The arguments of the previous paragraph can then be applied to this case as well. □

Theorem 3. *Algorithm GA is* $\frac{2r}{r-c}$*-competitive for 1S2L-F if* $0 \leq c < r$ *and* $a \geq t$.

Proof. We partition the time horizon $[0, \infty)$ into intervals (periods) that can be analyzed independently. Period i, for $1 < i < k$, is the interval $[t_{r'_i}, t_{r'_{i+1}})$. Period 1 is $[0, t_{r'_2})$, and period k is $[t_{r'_k}, \infty)$. (If $k = 1$, there is only a single period $[0, \infty)$.) Exactly one request in R', namely r'_i, starts in period i, for $1 \leq i \leq k$. We define $R'_i = \{r'_i\}$ for $1 \leq i \leq k$. Let R^*_i denote the set of requests accepted by OPT that start in period i, for $1 \leq i \leq k$.

For $1 < i \leq k$, r'_i starts at time $t_{r'_i}$ and the first request of R^*_i starts during the interval $[t_{r'_i}, t_{r'_{i+1}})$ (or the interval $[t_{r'_k}, \infty)$ if $i = k$). Furthermore, r'_1 is the first acceptable request in R, and so the first request of R^*_1 cannot start before r'_1. Hence, for all $1 \leq i \leq k$, the first request in R^*_i cannot start before the request r'_i.

We bound the competitive ratio of GA by analyzing each period independently. As $R' = \bigcup_i R'_i$ and $R^* = \bigcup_i R^*_i$, it is clear that $P_{R^*}/P_{R'} \leq \alpha$ follows if we can show that $P_{R^*_i}/P_{R'_i} \leq \alpha$ for all i, $1 \leq i \leq k$.

For all $1 \leq i \leq k$, as $R'_i = \{r_i\}$, we have $P_{R'_i} \in \{r, r - c\}$. It suffices to show $P_{R^*_i}/P_{R'_i} \leq 2r/(r - c)$ to prove the theorem. We claim that R^*_i contains at most two requests. Assume that R^*_i contains at least three requests. Let

r_j be the third request (in order of start time) in R_i^*. As the first request in R_i^* does not start before $t_{r_i'}$, we have $t_{r_j} \geq t_{r_i'} + 2t$. This means that r_j would be acceptable to GA after it has accepted r_i'. Therefore, GA accepts either r_j or another request starting before t_{r_j}, and that request becomes r_{i+1}'. Hence, there cannot be such a request r_j that starts in period i.

As we have shown that R_i^* contains at most two requests, we get that $P_{R_i^*} \leq 2r$. Since $P_{R_i'} \geq r - c$, we have $P_{R_i^*}/P_{R_i'} \leq 2r/(r-c)$. The theorem follows. □

3. Car Sharing with Variable Booking Times

In this section, we study the 1S2L-V problem. Recall that the booking time of a request r_i must satisfy $t_{r_i} - b_u \leq \tilde{t}_{r_i} \leq t_{r_i} - b_l$. First, we present three lower bound results, two for the case $c < r$ and one for the case $c = r$.

Theorem 4. *No deterministic algorithm for 1S2L-V can achieve a competitive ratio smaller than* $\frac{3r-c}{r-c}$ *if* $0 \leq c < r$ *and* $b_u > t$. *Furthermore, no randomized online algorithm can achieve a competitive ratio smaller than* $\frac{5r-c}{3r-c}$.

Proof. First, we give the lower bound for deterministic algorithms. Initially, the adversary releases the request $r_1 = (0, b_u, 1)$. We distinguish two cases.

Case 1: *ALG* accepts r_1 (with cost). The adversary releases requests $r_2 = (\varepsilon, t_{r_1} - \varepsilon, 1)$, $r_3 = (\varepsilon + t, t_{r_2} + t, 0)$ and $r_4 = (\varepsilon + 2t, t_{r_3} + t, 1)$, where $0 < \varepsilon < \min\{t, \frac{b_u - b_l}{2}\}$. *OPT* accepts r_2, r_3 and r_4. As they are in conflict with r_1, *ALG* cannot accept any of them, see also Fig. 2. We have $P_{R^*} = 3r - c$ and $P_{R'} = r - c$. Hence, $P_{R^*}/P_{R'} = \frac{3r-c}{r-c}$.

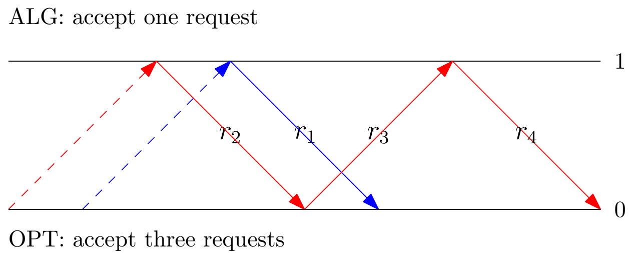


Figure 2: Case 1: $\frac{P_{R^*}}{P_{R'}} = \frac{3r-c}{r-c}$

Case 2: *ALG* does not accept request r_1. In this case, *OPT* accepts r_1. We have $P_{R^*} = r - c$, $P_{R'} = 0$, and hence $P_{R^*}/P_{R'} = \infty$.

This completes the proof of the lower bound for deterministic algorithms. To get a lower bound for randomized algorithms, we apply Yao's principle, in a similar way as in the proof of Theorem 1. We present request sequence $\sigma_1 = \{r_1\}$ with probability $p = \frac{2r}{3r-c}$ and request sequence $\sigma_2 = \{r_1, r_2, r_3, r_4\}$ with probability $1-p$, where r_1, r_2, r_3, r_4 are defined as above. Let $y(j)$ denote the resulting probability distribution. All deterministic online algorithms either accept r_1 (class ALG_1) or reject r_1 (class ALG_2). Algorithms in ALG_1 achieve profit $r-c$ on both request sequences. Algorithms in ALG_2 achieve profit 0 on σ_1 and profit at most $3r-c$ on σ_2, thus an expected profit of at most $(1-p)(3r-c) = r-c$. The profit of OPT is $r-c$ on σ_1 and $3r-c$ on σ_2, thus an expected profit of $(r-c)\frac{5r-c}{3r-c}$. See also Table 3. As we have

Table 3: Profits for σ_1 and σ_2 in the proof of Theorem 4

	σ_1	σ_2
ALG_1	$r-c$	$r-c$
ALG_2	0	$3r-c$
OPT	$r-c$	$3r-c$

$\mathbf{E}_{y(j)}[A(\sigma_j)] \le r-c$ and $\mathbf{E}_{y(j)}[OPT(\sigma_j)] = (r-c)\frac{5r-c}{3r-c}$, Yao's principle gives a lower bound of $\frac{5r-c}{3r-c}$ on the competitive ratio of randomized algorithms. $\square$

Theorem 5. *No deterministic algorithm for 1S2L-V can have a competitive ratio smaller than* 3 *if* $0 \le c \le r$ *and* $b_u \le t$. *Furthermore, no randomized online algorithm can achieve a competitive ratio smaller than* $\frac{5}{3}$.

PROOF. First, we give the lower bound for deterministic algorithms. Initially, the adversary releases the request $r_1 = (0, b_u, 0)$. We distinguish two cases.

Case 1: ALG accepts r_1 (without cost). The adversary releases requests $r_2 = (\varepsilon, t_{r_1} - \varepsilon, 0)$, $r_3 = (\varepsilon + t, t_{r_2} + t, 1)$ and $r_4 = (\varepsilon + 2t, t_{r_3} + t, 0)$, where $0 < \varepsilon < \min\{t, \frac{b_u - b_l}{2}\}$. OPT accepts r_2, r_3 and r_4. As they are in conflict with r_1, ALG cannot accept any of them. We have $P_{R^*} = 3r$ and $P_{R'} = r$. Hence, $P_{R^*}/P_{R'} = 3$.

Case 2: ALG does not accept request r_1. In this case, OPT accepts r_1. We have $P_{R^*} = r$, $P_{R'} = 0$, and hence $P_{R^*}/P_{R'} = \infty$.

This completes the proof of the lower bound for deterministic algorithms. To get a lower bound for randomized algorithms, we use Yao's principle and present request sequence $\sigma_1 = \{r_1\}$ with probability $p = \frac{2}{3}$ and request sequence $\sigma_2 = \{r_1, r_2, r_3, r_4\}$ with probability $1 - p = \frac{1}{3}$, where r_1, r_2, r_3, r_4 are defined as above. Let $y(j)$ denote the resulting probability distribution. All deterministic online algorithms either accept r_1 (class ALG_1) or reject r_1 (class ALG_2). Algorithms in ALG_1 achieve profit r on both request sequences. Algorithms in ALG_2 achieve profit 0 on σ_1 and profit at most $3r$ on σ_2, thus an expected profit of at most $(1-p)3r = r$. The profit of OPT is r on σ_1 and $3r$ on σ_2, thus an expected profit of $\frac{5}{3}r$. See also Table 4.As

Table 4: Profits for σ_1 and σ_2 in the proof of Theorem 5

	σ_1	σ_2
ALG_1	r	r
ALG_2	0	$3r$
OPT	r	$3r$

we have $\mathbf{E}_{y(j)}[A(\sigma_j)] \leq r$ and $\mathbf{E}_{y(j)}[OPT(\sigma_j)] = \frac{5}{3}r$, Yao's principle gives a lower bound of $\frac{5}{3}$ on the competitive ratio of randomized algorithms. □

From Theorems 1 and 5 we can conclude that no deterministic algorithm for 1S2L-V can have competitive ratio smaller than $\max\{\frac{2r}{r-c}, 3\}$ if $0 \leq c < r$ and $b_u = t$.

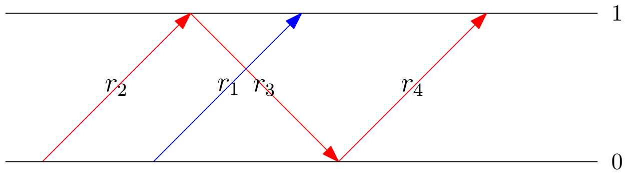
Theorem 6. *No deterministic algorithm for 1S2L-V can achieve a competitive ratio smaller than $1 + 2\lceil \frac{b_u - b_l}{2t} \rceil$ if $c = r$. Furthermore, no randomized online algorithm can achieve a competitive ratio smaller than $\frac{5}{3}$.*

Proof. The lower bound for randomized algorithms follows from Theorem 5. In the following we present the lower bound for deterministic algorithms. Let ALG be an arbitrary online algorithm, and let OPT be an optimal scheduler. We distinguish two cases based on the value of $b_u - b_l$.

Case 1: $0 < b_u - b_l \leq 2t$. We need to show that the competitive ratio is at least 3. Define four requests as follows: $r_1 = (\frac{b_u+b_l}{2} + t, b_u + b_l + t, 0)$, $r_2 = (\frac{2b_u+b_l}{3} + t, b_l + \frac{2b_u+b_l}{3} + t, 0)$, $r_3 = (\frac{2b_u+b_l}{3} + 2t, b_l + \frac{2b_u+b_l}{3} + 2t, 1)$, $r_4 = (\frac{2b_u+b_l}{3} + 3t, b_l + \frac{2b_u+b_l}{3} + 3t, 0)$. Note that a server can accept either r_1, or all of r_2, r_3, r_4. Furthermore, r_2 is released after r_1 but starts earlier.

Initially, the adversary releases r_1. There are two sub-cases.

ALG: accept one request



OPT: accept three requests

Figure 3: Illustration of R^* and R' in Case 1.1

Case 1.1: *ALG* accepts r_1. The adversary releases r_2, r_3 and r_4. *OPT* accepts r_2, r_3, r_4 without cost (see Fig. 3 for an illustration), so we have $P_{R^*} = 3r$ and $P_{R'} = r$, showing that $P_{R^*}/P_{R'} = 3$.

Case 1.2: *ALG* does not accept request r_1. *OPT* accepts r_1. We have $P_{R^*} = r$ and $P_{R'} = 0$, and hence $P_{R^*}/P_{R'} = \infty$.

The lower bound of 3 follows.

Case 2: $2t < b_u - b_l$. Let $n = \lceil \frac{b_u - b_l}{2t} \rceil - 1$. Choose values ε_i for $1 \le i \le n+2$ satisfying $0 \le \varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_{n+1} < \varepsilon_{n+2} < \min\{t, b_u - b_l - 2tn\}$.

Initially, the adversary releases the request sequence R_1 consisting of the following requests: $r_1 = (\varepsilon_1, b_l + \varepsilon_{n+2} + t, 1), r_2 = (\varepsilon_2, b_l + \varepsilon_{n+2} + 3t, 1), \ldots, r_i = (\varepsilon_i, b_l + \varepsilon_{n+2} + (2i-1)t, 1), \ldots, r_n = (\varepsilon_n, b_l + \varepsilon_{n+2} + (2n-1)t, 1)$ and $r_{n+1} = (\varepsilon_{n+1}, b_u + \varepsilon_{n+1}, 0)$. Note that $b_l \le b_l + \varepsilon_{n+2} + (2i-1)t \le b_u$ for all $1 \le i \le n$. There are three sub-cases.

Case 2.1: *ALG* rejects all the requests of R_1. In this case, *OPT* accepts the request r_{n+1}. We have $P_{R^*} = r$ and $P_{R'} = 0$, yielding $P_{R^*}/P_{R'} = \infty$.

Case 2.2: The first request accepted by *ALG* is r_i for some i with $1 \le i \le n$. In this case, the adversary does not release the remaining requests of R_1. Instead, it releases only one final request $r_f = (\varepsilon_{i+1}, b_l + (2i-1)t, 0)$. *ALG* cannot accept r_f as it is in conflict with r_i. *OPT* accepts r_f. We have $P_{R^*} = r$ and $P_{R'} = r - c = 0$, hence $P_{R^*}/P_{R'} = \infty$.

Case 2.3: The first request accepted by *ALG* is r_{n+1}. The adversary then releases the request sequence R_2 consisting of the following requests: $r_{n+1+1} = (\varepsilon_{n+2}, b_l + \varepsilon_{n+2}, 0)$, $r_{n+1+2} = (\varepsilon_{n+2}, b_l + \varepsilon_{n+2} + 2t, 0)$, $\ldots$, $r_{n+1+i} = (\varepsilon_{n+2}, b_l + \varepsilon_{n+2} + 2(i-1)t, 0)$, $\ldots$, $r_{n+1+n} = (\varepsilon_{n+2}, b_l + \varepsilon_{n+2} + 2(n-1)t, 0)$. After this, the adversary releases the request sequence R_3 consisting of three more requests: $r_{2n+1+1} = (\varepsilon_{n+2}, b_u, 0)$, $r_{2n+1+2} = (\varepsilon_{n+2} + t, b_u + t, 1)$, $r_{2n+1+3} = (\varepsilon_{n+2} + 2t, b_u + 2t, 0)$. *ALG* cannot accept any requests of R_3 as they all conflict with r_{n+1} (see Fig. 4 for an illustration). *ALG* can accept any number of requests of R_2, but since they all have pick-up location 0 (as does r_{n+1}), its

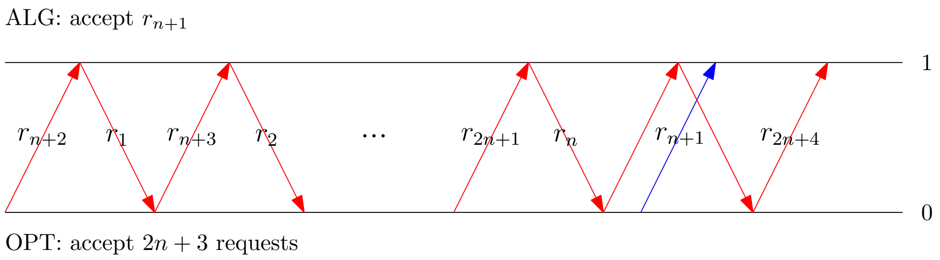


Figure 4: Illustration of R^* and R' in Case 2.3

total profit will be $P_{R'} = r$. OPT accepts all requests of R_1 except r_{n+1}, and all requests of R_2 and R_3. We have $P_{R^*} = (2n+3)r = (2\lceil \frac{b_u - b_l}{2t} \rceil + 1)r$. Hence, $P_{R^*}/P_{R'} = 1 + 2\lceil \frac{b_u - b_l}{2t} \rceil$.

The claimed lower bound of $1 + 2\lceil \frac{b_u - b_l}{2t} \rceil$ follows. $\square$

We now turn to upper bounds and analyze Algorithm GA (which was presented as Algorithm 1 in Section 2) for the 1S2L-V problem. Denote the set of requests accepted by OPT by R^* and the set of requests accepted by GA as R'. The server of OPT is referred to as s^*, and the server of GA as s'. Let $R' = \{r'_1, \ldots, r'_k\}$, with the requests indexed in order of increasing start time. For $1 \le i \le k$, let $R'_i = \{r'_i\}$. We partition the time horizon $[0, \infty)$ into intervals (periods) that can be analyzed independently. The partition differs for GA and OPT, so we refer to *GA periods* and *OPT periods*. GA period i is the interval $[0, t_{r'_2})$ if $i = 1$, the interval $[t_{r'_k}, \infty)$ if $i = k$, and the interval $[t_{r'_i}, t_{r'_{i+1}})$ if $1 < i < k$. Note that R'_i consists of the only request in R' that starts in GA period i.

For $1 \le i \le k$, define $\hat{t}_{r'_i}$ to be the first time when the optimal server s^* is at location $\dot{p}_{r'_i}$ at or after time $\dot{t}_{r'_i}$, or ∞ if s^* never reaches $\dot{p}_{r'_i}$ from time $\dot{t}_{r'_i}$ onward. Now, define OPT period i to be the interval $[0, \hat{t}_{r'_1})$ if $i = 1$, the interval $[\hat{t}_{r'_{k-1}}, \infty)$ if $i = k$, and the interval $[\hat{t}_{r'_{i-1}}, \hat{t}_{r'_i})$ if $1 < i < k$. For $1 \le i \le k$, let R^*_i be the set of requests accepted by OPT that start during OPT period i. If $\hat{t}_{r'_i} \le \hat{t}_{r'_{i-1}}$, OPT period i is empty and $R^*_i = \emptyset$. The j^{th} (in order of start times) request of R^*_i is denoted by $R^*_i(j)$.

We will compare the profit $P_{R'_i}$ that GA accrues in GA period i with the profit $P_{R^*_i}$ that OPT accrues in OPT period i. We can again analyze each period independently: If we can show that $P_{R^*_i}/P_{R'_i} \le \alpha$ for all i, this implies that $P_{R^*}/P_{R'} \le \alpha$. We first state two observations.

Observation 1. *For all i, s' is at $\dot{p}_{r'_i}$ at time $\dot{t}_{r'_i}$, and s^* is at $\dot{p}_{r'_i}$ at time $\hat{t}_{r'_i}$, and $\hat{t}_{r'_i} \ge \dot{t}_{r'_i}$.*

Observation 2. *If $c = r$, by definition of Algorithm GA (Algorithm 1), for $1 < i \leq k$ we have that $\tilde{t}_{r'_{i-1}} \leq \tilde{t}_{r'_i}$ (otherwise, $P_{R_i^{GA} \cup \{r'_i\}} - P_{R_i^{GA}} = 0$ and r'_i would be rejected).*

Theorems 7–9 consider the setting with $0 \leq c < r$. In the proofs, we compare the maximum profit gained by OPT in OPT period i with the profit gained by GA in GA period i. For the case $b_u < t$, GA and OPT only accept requests without cost, and we can show that OPT can accept at most three requests in each OPT period, giving a competitive ratio of 3 (Theorem 7). If $b_u = t$, the additional possibility arises that GA accepts a request with cost, while OPT accepts two requests without cost in the corresponding OPT period, giving ratio $\max\{3, \frac{2r}{r-c}\}$ (Theorem 8). If $b_u > t$, it additionally becomes possible that GA accepts a request with cost, while OPT accepts a request starting earlier with cost and then two further requests without cost, giving a ratio of $\frac{3r-c}{r-c}$ (Theorem 9). Finally, the setting with $r = c$ is considered in Theorem 10 and is the most complex to analyze. For the case where GA accepts a single request r_i (without cost), OPT can accept a number of pairs of requests from 0 to 1 and from 1 to 0 that start before the end time of r_i, and we can bound the number of those pairs of requests depending on the size $b_u - b_l$ of the booking horizon. For the general case where GA accepts an arbitrary number of requests, we give a proof by induction: We use the argument sketched above to handle the final request r_i accepted by GA, and apply the induction hypothesis to the smaller instance without r_i (and without any requests released later that GA could accept instead of r_i). The resulting competitive ratio is $1 + 2\lceil \frac{b_u - b_l}{2t} \rceil$.

Theorem 7. *Algorithm GA is 3-competitive for 1S2L-V if $0 < b_u < t$ and $0 \leq c < r$.*

Proof. Due to $0 < b_u < t$, all requests of R_i^* and R'_i must be accepted without cost because the request arrival is too late to serve a request with cost (recall that we forbid unprompted moves by OPT).

First, consider period $i = 1$. OPT cannot accept any request that is released during the time interval $[0, \tilde{t}_{r'_1})$, because otherwise such a request accepted by OPT could be accepted by GA instead of r'_1. Thus $\tilde{t}_{R_1^*(1)} \geq \tilde{t}_{r'_1}$, and hence $t_{R_1^*(1)} \geq \tilde{t}_{r'_1} \geq t_{r'_1} - b_u$. By Observation 1 and because OPT does not accept any request with cost, $\dot{p}_{R_i^*(1)} = \dot{p}_{r'_1}$, $\dot{p}_{R_1^*(2)} = p_{r'_1}$, and $\dot{p}_{R_1^*(3)} = \dot{p}_{r'_1}$. Hence, s^* is at $\dot{p}_{r'_1}$ at time $\dot{t}_{R_i^*(3)}$, which is not before $\dot{t}_{r'_1}$ (because $t_{R_1^*(3)} \geq t_{R_1^*(1)} + 2t \geq t_{r'_1} - b_u + 2t > t_{r'_1} + t$). Therefore, $|R_1^*| \leq 3$. Thus, $P_{R_1^*}/P_{R'_1} \leq 3$.

For $1 < i \leq k$, we distinguish the following cases in order to bound $P_{R_i^*}/P_{R_i'}$. As $R_i' = \{r_i'\}$, $P_{R_i'} = r$. We need to show that $P_{R_i^*} \leq 3r$.

Case 1: $\hat{t}_{r'_{i-1}} < \hat{t}_{r'_i}$ and $\hat{t}_{r'_{i-1}} < t_{r'_i}$. Assume that $|R_i^*| \geq 3$. (Otherwise, there is nothing to show.) By Observation 1 and because OPT does not accept any request with cost, $\dot{p}_{R_i^*(1)} = \dot{p}_{r_i'}$, $\dot{p}_{R_i^*(2)} = p_{r_i'}$, and $\dot{p}_{R_i^*(3)} = \dot{p}_{r_i'}$. If $t_{R_i^*(1)} \geq t_{r_i'}$, we have $\hat{t}_{r_i'} = \dot{t}_{R_i^*(1)}$ and thus $|R_i^*| = 1$. Therefore, we must have $t_{R_i^*(1)} < t_{r_i'}$. Observe that OPT can only accept requests with pick-up location $p_{r_i'}$ that start before $t_{r_i'}$ if they start after $t_{r_i'} - b_u > t_{r_i'} - t$. Otherwise, such a request accepted by OPT would arrive before r_i' and so it would be accepted by GA instead of r_i'. So we have $\dot{t}_{R_i^*(3)} \geq t_{R_i^*(1)} + 3t \geq t_{r_i'} - b_u + 3t > \dot{t}_{r_i'}$. When OPT finishes serving the third request of R_i^*, s^* is at $\dot{p}_{r_i'}$, and this happens at a time after $\dot{t}_{r_i'}$ (see Fig. 5 for an illustration). Therefore, this time is $\hat{t}_{r_i'}$ and thus the end of OPT period i, and R_i^* cannot contain any further requests. (If $i = k$, the end of OPT period i is ∞, but any further request accepted by OPT could also be accepted by GA, a contradiction.) We have shown $|R_i^*| \leq 3$, as required. Hence $P_{R_i^*}/P_{R_i'} \leq 3$.

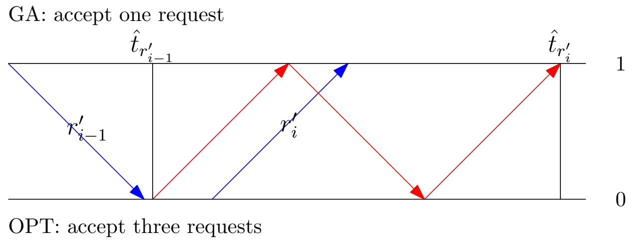


Figure 5: Example configuration for R_i^* and R_i' in Case 1

Case 2: $\hat{t}_{r'_{i-1}} < \hat{t}_{r'_i}$ and $\hat{t}_{r'_{i-1}} \geq t_{r'_i}$. We claim that $|R_i^*| \leq 1$ and argue as follows. Because $\hat{t}_{r'_{i-1}} \geq t_{r'_i}$, OPT can accept at most one request in OPT period i: When s^* finishes serving the first request in R_i^*, s^* is located at $\dot{p}_{r_i'}$, and the time when this happens becomes $\hat{t}_{r_i'}$ and thus the end of OPT period i. (If $i = k$, we can argue as in Case 1 that OPT cannot accept any further requests.) Hence, $P_{R_i^*}/P_{R_i'} \leq 1 < 3$.

Case 3: $\hat{t}_{r'_{i-1}} \geq \hat{t}_{r'_i}$. As OPT period i is empty by definition, we have $R_i^* = \emptyset$ and hence $P_{R_i^*} = 0$. Thus, $P_{R_i^*}/P_{R_i'} < 3$.

Because $P_{R_i^*}/P_{R_i'} \leq 3$ holds for all $1 \leq i \leq k$, we have $P_{R^*}/P_{R'} \leq 3$. This proves the theorem. □

Theorem 8. *Algorithm GA is* $\max\{\frac{2r}{r-c}, 3\}$*-competitive for 1S2L-V if* $0 \leq c < r$ *and* $b_u = t$.

Proof. First, consider period $i = 1$. *OPT* cannot accept any request that is released during the time interval $[0, \tilde{t}_{r'_1})$, because otherwise such a request accepted by *OPT* could be accepted by GA instead of r'_1. Thus $\tilde{t}_{R_1^*(1)} \geq \tilde{t}_{r'_1}$, and hence $t_{R_1^*(1)} \geq \tilde{t}_{R_1^*(1)} \geq \tilde{t}_{r'_1} \geq t_{r'_1} - b_u = t_{r'_1} - t$ (as $b_u = t$). Furthermore, if $p_{R_1^*(1)} = 1$, $t_{R_1^*(1)} \geq \tilde{t}_{R_1^*(1)} + t$ (recall that *OPT* is not allowed to make unprompted moves), and hence $t_{R_1^*(1)} \geq \tilde{t}_{r'_1} + t \geq t_{r'_1}$ (note that the booking interval of r'_1, $t_{r'_1} - \tilde{t}_{r'_1}$, is not greater than t). There are two sub-cases based on the pick-up location of r'_1.

(1) $p_{r'_1} = 0$. Note that $P_{R'_1} = r$. Since $t_{R_1^*(1)} \geq t_{r'_1} - t$, $t_{R_1^*(3)} \geq t_{r'_1} + t = \dot{t}_{r'_1}$. Hence, s^* is at $\dot{p}_{R_1^*(3)}$ at time $\dot{t}_{R_1^*(3)}$, which is not before $\dot{t}_{r'_1}$, and at $p_{R_1^*(3)}$ at time $t_{R_1^*(3)}$, which is not before $\dot{t}_{r'_1}$. Therefore, $|R_1^*| \leq 3$. Thus, $P_{R_1^*}/P_{R'_1} \leq 3$.

(2) $p_{r'_1} = 1$. Note that $P_{R'_1} = r - c$. If $p_{R_1^*(1)} = 0$, then s^* is at $\dot{p}_{r'_1}$ $(= 1)$ at time $\dot{t}_{R_i^*(1)}$ $(\geq t_{r'_1})$. We claim that *OPT* can accept at most one more request in *OPT* period 1: if $p_{R_1^*(2)} = 1$, then s^* is at 0 at time $\dot{t}_{R_i^*(2)}$ $(\geq t_{r'_1} + t)$, which is not before $\dot{t}_{r'_1}$; if $p_{R_1^*(2)} = 0$, then s^* is at 0 at time $t_{R_i^*(2)}$ $(\geq t_{r'_1} + t)$, which is not before $\dot{t}_{r'_1}$. Therefore, $|R_1^*| \leq 2$. Hence, $P_{R_1^*}/P_{R'_1} \leq \frac{2r}{r-c}$. If $p_{R_1^*(1)} = 1$, then $\dot{t}_{R_1^*(1)} \geq \dot{t}_{r'_1}$. Hence, s^* is at $\dot{p}_{r'_1}$ at time $\dot{t}_{R_i^*(1)}$, which is not before $\dot{t}_{r'_1}$. Therefore, $|R_1^*| \leq 1$. Thus, $P_{R_1^*}/P_{R'_1} \leq \frac{r-c}{r-c} = 1$.

For $1 < i \leq k$, we distinguish the following cases in order to bound $P_{R_i^*}/P_{R'_i}$. As $R'_i = \{r'_i\}$, $P_{R'_i} = r$ or $P_{R'_i} = r - c$. We need to show that $P_{R_i^*} \leq 3r$ when $P_{R'_i} = r$, and $P_{R_i^*} \leq 2r$ when $P_{R'_i} = r - c$.

R_i^* cannot contain any request that is released before $\tilde{t}_{r'_i}$, because otherwise such a request accepted by *OPT* could be accepted by GA instead of r'_i. Thus $\tilde{t}_{R_i^*(1)} \geq \tilde{t}_{r'_i}$, and hence $t_{R_i^*(1)} \geq \tilde{t}_{R_i^*(1)} \geq \tilde{t}_{r'_i} \geq t_{r'_i} - b_u = t_{r'_i} - t$. Furthermore, if $p_{R_i^*(1)} = p_{r'_{i-1}}$, $t_{R_i^*(1)} \geq \tilde{t}_{R_i^*(1)} + t$ (recall that *OPT* is not allowed to make unprompted moves), and hence $t_{R_i^*(1)} \geq \tilde{t}_{r'_i} + t \geq t_{r'_i}$ (as the booking interval of r'_i, $t_{r'_i} - \tilde{t}_{r'_i}$, is not greater than t).

Case 1: $\hat{t}_{r'_{i-1}} < \hat{t}_{r'_i}$ and $\hat{t}_{r'_{i-1}} < t_{r'_i}$. There are two sub-cases based on the pick-up location of r'_i.

(1) $p_{r'_i} = \dot{p}_{r'_{i-1}}$. Note that $P_{R'_i} = r$. Since $t_{R_i^*(1)} \geq t_{r'_i} - t$, $t_{R_i^*(3)} \geq t_{r'_i} + t = \dot{t}_{r'_i}$. Hence, s^* is at $\dot{p}_{R_i^*(3)}$ at time $\dot{t}_{R_i^*(3)}$, which is not before $\dot{t}_{r'_i}$, and at $p_{R_i^*(3)}$ at time $t_{R_i^*(3)}$, which is not before $\dot{t}_{r'_i}$. Therefore, $|R_i^*| \leq 3$. Thus, $P_{R_i^*}/P_{R'_i} \leq 3$.

(2) $p_{r'_i} = p_{r'_{i-1}}$. Note that $P_{R'_i} = r - c$. If $p_{R_i^*(1)} = \dot{p}_{r'_{i-1}}$ $(= \dot{p}_{r'_i})$, then s^* is at $\dot{p}_{R_i^*(1)}$ $(= p_{r'_i})$ at time $\dot{t}_{R_i^*(1)}$ $(\geq t_{r'_i})$. We claim that *OPT* can accept at most one more request in *OPT* period i: if $p_{R_i^*(2)} = p_{R_i^*(1)}$, then s^* is at $p_{R_i^*(2)}$

(=$\dot{p}_{r'_i}$) at time $\dot{t}_{R^*_i(2)}$ ($\geq t_{r'_i} + t$), which is not before $\dot{t}_{r'_i}$; if $p_{R^*_1(2)} = \dot{p}_{R^*_i(1)}$, then s^* is at $\dot{p}_{R^*_i(2)}$ (=$\dot{p}_{r'_i}$) at time $\dot{t}_{R^*_i(2)}$ ($\geq t_{r'_i} + t$), which is not before $\dot{t}_{r'_i}$. Therefore, $|R^*_i| \leq 2$. Hence, $P_{R^*_i}/P_{R'_i} \leq \frac{2r}{r-c}$. If $p_{R^*_i(1)} = p_{r'_{i-1}}$, then $t_{R^*_i(1)} \geq t_{r'_i}$, $t_{R^*_i(2)} \geq t_{r'_i} + t = \dot{t}_{r'_i}$. Hence, s^* is at $\dot{p}_{R^*_i(2)}$ at time $\dot{t}_{R^*_i(2)}$, which is not before $\dot{t}_{r'_i}$, and at $p_{R^*_i(2)}$ at time $t_{R^*_i(2)}$, which is not before $\dot{t}_{r'_i}$. Therefore, $|R^*_1| \leq 2$. Thus, $P_{R^*_i}/P_{R'_i} \leq \frac{2r}{r-c}$.

If $i = k$, the end of OPT period i is ∞, but any further request accepted by OPT could also be accepted by GA, a contradiction.

Case 2: $\hat{t}_{r'_{i-1}} < \hat{t}_{r'_i}$ and $\hat{t}_{r'_{i-1}} \geq t_{r'_i}$. Observe that $t_{R^*_i(1)} \geq t_{r'_i}$. We claim that $|R^*_i| \leq 2$ and argue as follows. OPT can accept at most two requests in OPT period i: When s^* starts serving the second request in R^*_i, s^* is located at $p_{R^*_i(2)}$ at $t_{R^*_i(2)}$ ($\geq t_{r'_i} + t$), so that time becomes $\hat{t}_{r'_i}$ and thus the end of OPT period i if $p_{R^*_i(2)} = \dot{p}_{r'_i}$; when s^* finishes serving the second request in R^*_i, s^* is located at $\dot{p}_{R^*_i(2)}$ at $\dot{t}_{R^*_i(2)}$ ($\geq t_{r'_i} + 2t$), so that time becomes $\hat{t}_{r'_i}$ and thus the end of OPT period i if $\dot{p}_{R^*_i(2)} = \dot{p}_{r'_i}$. (If $i = k$, we can argue as in Case 1 that OPT cannot accept any further requests.) Since $P_{R'_i} \geq r - c$, $P_{R^*_i}/P_{R'_i} \leq \frac{2r}{r-c}$.

Case 3: $\hat{t}_{r'_{i-1}} \geq \hat{t}_{r'_i}$. As OPT period i is empty by definition, we have $R^*_i = \emptyset$ and hence $P_{R^*_i} = 0$. Thus, $P_{R^*_i}/P_{R'_i} < \max\{\frac{2r}{r-c}, 3\}$.

Because $P_{R^*_i}/P_{R'_i} \leq \max\{\frac{2r}{r-c}, 3\}$ holds for all $1 \leq i \leq k$, we have $P_{R^*}/P_{R'} \leq \max\{\frac{2r}{r-c}, 3\}$. This proves the theorem. □

Theorem 9. *Algorithm GA is $\frac{3r-c}{r-c}$-competitive for 1S2L-V if* $0 \leq c < r$ *and* $b_u > t$.

Proof. We show that $P_{R^*_i}/P_{R'_i} \leq (3r - c)/(r - c)$ for all $1 \leq i \leq k$.

First, consider $i = 1$. We have $P_{R'_1} \geq r - c$. If $|R^*_1| \leq 2$, then $P_{R^*_1} \leq 2r < 3r - c$ and $P_{R^*_1}/P_{R'_1} \leq (3r-c)/(r-c)$ as claimed. Thus, assume $|R^*_1| \geq 3$ from now on. OPT does not accept any request with pick-up location $\dot{p}_{r'_1}$ that starts during period $[0, t_{r'_1} - t]$, and it does not accept any request with pick-up location $p_{r'_1}$ during period $[0, t_{r'_1} - 2t]$. Otherwise, such a request accepted by OPT could be accepted by GA, a contradiction to r'_1 being the request with earliest start time in R'. If $p_{R^*_1(1)} = p_{r'_1}$ and the first three requests in R^*_1 have alternating pick-up locations, $\dot{t}_{R'_1(3)} > \dot{t}_{r'_1}$ follows from $t_{R^*_1(1)} > t_{r'_1} - 2t$. As $\dot{p}_{P_{R'_1}(3)} = \dot{p}_{r'_1}$, we have $\hat{t}_{r'_1} = \dot{t}_{R'_1(3)}$ and thus $|R^*_1| = 3$. If $p_{r'_1} = 0$, we have $P_{R'_1} = r$ and $P_{R^*_1} = 3r$, giving $P_{R^*_1}/P_{R'_1} \leq 3 < (3r - c)/(r - c)$. If $p_{r'_1} = 1$, we have $P_{R'_1} = r - c$ and $P_{R^*_1} = 3r - c$, giving $P_{R^*_1}/P_{R'_1} = (3r - c)/(r - c)$.

If $p_{R_1^*(1)} = p_{r_1'}$ and the first three requests in R_1^* do not have alternating pick-up locations, the start time of one of the requests is at least t units of time after the end time of the preceding request, and thus $\dot{t}_{R_1'(3)} > \dot{t}_{r_1'} + t$. As s^* is at $\dot{p}_{r_1'}$ either at time $t_{R_1^*(3)}$ or at time $\dot{t}_{R_1^*(3)}$, and as both times are after $\dot{t}_{r_1'}$, OPT period 1 ends before a fourth request in R_1^* could start. Thus, $|R_1^*| \leq 3$ and $P_{R_1^*} \leq 3r - c$, since serving two consecutive requests with the same pick-up location means accepting the second of these requests with cost. If $p_{R_1^*(1)} \neq p_{r_1'}$ and the first two requests in R_1^* have alternating pick-up locations, $\dot{t}_{R_1^*(2)} > \dot{t}_{r_1'}$ follows from $t_{R_1^*(1)} > t_{r_1'} - t$. As $\dot{p}_{R_1^*(2)} = \dot{p}_{r_1'}$, we have $\hat{t}_{r_1'} = \dot{t}_{R_1^*(2)}$ and thus $|R_1^*| = 2$ and $P_{R_1^*} \leq 2r$, showing $P_{R_1^*}/P_{R_1'} \leq 2r/(r-c) < (3r-c)/(r-c)$. If $p_{R_1^*(1)} \neq p_{r_1'}$ and the first two requests in R_1^* have the same pick-up location, the start time of the second request is at least t units of time after the end time of the first request, and thus $\dot{t}_{R_1^*(2)} > \dot{t}_{r_1'} + t$. As s^* is at $\dot{p}_{r_1'}$ either at time $t_{R_1^*(2)}$ or at time $\dot{t}_{R_1^*(2)}$, and as both times are after $\dot{t}_{r_1'}$, OPT period 1 ends before a third request in R_1^* could start. Thus, $|R_1^*| \leq 2$ and $P_{R_1^*} \leq 2r - c$, yielding $P_{R_1^*}/P_{R_1'} \leq \frac{2r-c}{r-c} < \frac{3r-c}{r-c}$.

Hence $P_{R_1^*}/P_{R_1'} \leq \frac{3r-c}{r-c}$ in all cases.

For $1 < i \leq k$, we distinguish the following cases.

Case 1: r_i' is accepted without cost. We have $P_{R_i'} = r$ and $\dot{p}_{r_{i-1}'} = p_{r_i'}$, and we know that s^* is at $p_{r_i'}$ at time $\hat{t}_{r_{i-1}'}$.

Case 1.1: $\hat{t}_{r_{i-1}'} < \hat{t}_{r_i'}$ and $\hat{t}_{r_{i-1}'} < t_{r_i'}$. We show $P_{R_i^*} \leq 3r$ by considering the following sub-cases.

Case 1.1.1: OPT accepts $R_i^*(1)$ with cost. We have $p_{R_i^*(1)} = p_{r_{i-1}'}$. See Fig. 6 for an illustration. Because any request with pick-up location $p_{r_{i-1}'}$ that starts during period $[\dot{t}_{r_{i-1}'} + t, t_{r_i'} - t]$ could be accepted by GA, we have $t_{R_i^*(1)} > t_{r_i'} - t$. Thus $\dot{t}_{R_i^*(1)} + t > \dot{t}_{r_i'}$. R_i^* can contain at most one request that starts from time $\dot{t}_{R_i^*(1)}$ onward during period $[\hat{t}_{r_{i-1}'}, \hat{t}_{r_i'})$ because any more requests would be assigned to R_{i+1}^* (or, if $i = k$, any more requests could be accepted by GA, a contradiction to r_k' being the last request in R'). Therefore we have $P_{R_i^*} \leq 2r - c$. Hence $P_{R_i^*}/P_{R_i'} \leq \frac{2r-c}{r} < \frac{3r-c}{r-c}$.

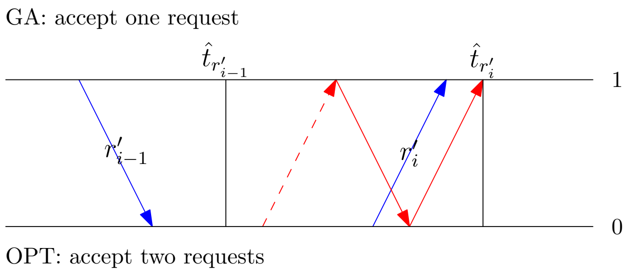


Figure 6: Illustration of R^*_i and R'_i in Case 1.1.1

Case 1.1.2: OPT accepts $R^*_i(1)$ without cost. We have $p_{R^*_i(1)} = \dot{p}_{r'_{i-1}}$. If $|R^*_i| \leq 2$, we have $P_{R^*_i} \leq 2r < 3r$, as required. Assume $|R^*_i| \geq 3$. See Fig. 7 for an illustration. Because any request with pick-up location $\dot{p}_{r'_{i-1}}$ that starts during period $[\dot{t}_{r'_{i-1}}, t_{r'_i} - 2t]$ could be accepted by GA, we have $t_{R^*_i(1)} > t_{r'_i} - 2t$. Thus $\dot{t}_{R^*_i(1)} + 2t > \dot{t}_{r'_i}$. If the first three requests in R^*_i have alternating pick-up locations, we have $\hat{t}_{r'_i} \leq \dot{t}_{R^*_i(3)}$, and hence R^*_i contains no more than three requests. If the first three requests in R^*_i do not have alternating pick-up locations, we have $\dot{t}_{R^*_i(3)} \geq \dot{t}_{R^*_i(1)} + 3t > \dot{t}_{r'_i} + t$. As s^* is at $\dot{p}_{r'_i}$ either at time $t_{R^*_i(3)}$ or at time $\dot{t}_{R^*_i(3)}$, $\hat{t}_{r'_i}$ cannot be later than that time. Therefore, $|R^*_i| \leq 3$. (If $i = k$, observe in addition that OPT cannot accept any request starting after $\hat{t}_{r'_i}$ as any such request accepted by OPT could also be accepted by GA.) We have $P_{R^*_i} \leq 3r$. Hence, $P_{R^*_i}/P_{R'_i} \leq \frac{3r}{r} < \frac{3r-c}{r-c}$.

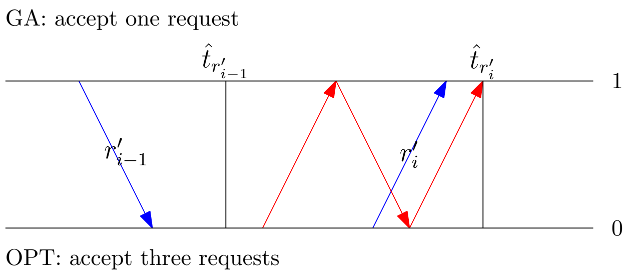


Figure 7: Illustration of R^*_i and R'_i in Case 1.1.2

Case 1.2: $\hat{t}_{r'_{i-1}} < \hat{t}_{r'_i}$ and $\hat{t}_{r'_{i-1}} \geq t_{r'_i}$. We claim $|R^*_i| \leq 1$ and argue as follows. Based on the definition of $\hat{t}_{r'_{i-1}}$, s^* is at $\dot{p}_{r'_{i-1}}$ at time $\hat{t}_{r'_{i-1}}$. Because $\hat{t}_{r'_{i-1}} \geq t_{r'_i}$, s^* can only accept one request without cost or move to $\dot{p}_{r'_i}$ for serving a request with cost during OPT i period $[\hat{t}_{r'_{i-1}}, \hat{t}_{r'_i})$. Hence $P_{R^*_i}/P_{R'_i} \leq \frac{r}{r} < \frac{3r-c}{r-c}$.

Case 1.3: $\hat{t}_{r'_{i-1}} \geq \hat{t}_{r'_i}$. As the OPT period i is empty by definition, $R^*_i = \emptyset$

and $P_{R_i^*} = 0$. Hence $P_{R_i^*}/P_{R_i'} < \frac{3r-c}{r-c}$.

Case 2: r_i' is accepted with cost. We have $P_{R_i'} = r - c$ and $p_{r_{i-1}'} = p_{r_i'}$. s^* is at $\dot{p}_{r_i'}$ at time $\hat{t}_{r_{i-1}'}$.

Case 2.1: $\hat{t}_{r_{i-1}'} < \hat{t}_{r_i'}$ and $\hat{t}_{r_{i-1}'} < t_{r_i'}$. We show $P_{R_i^*} \leq 3r-c$ by considering the following sub-cases.

Case 2.1.1: *OPT* accepts $R_i^*(1)$ with cost. Note that $p_{R_i^*(1)} = p_{r_{i-1}'}$. See Fig. 8 for an illustration. If $|R_i^*| \leq 2$, we have $P_{R_i^*} \leq 2r - c < 3r - c$ as required. Assume $|R_i^*| \geq 3$. Because any request with pick-up location $p_{r_{i-1}'}$ that starts during period $[\dot{t}_{r_{i-1}'} + t, t_{r_i'} - 2t]$ could be accepted by GA, we have $t_{R_i^*(1)} > t_{r_i'} - 2t$. If the first three requests in R_i^* have alternating pick-up locations, we have $\hat{t}_{r_i'} \leq t_{R_i^*(3)}$ (because $\dot{t}_{R_i^*(1)} + 2t > \dot{t}_{r_i'}$), and hence R_i^* contains no more than three requests. If the first three requests in R_i^* do not have alternating pick-up locations, we have $\dot{t}_{R_i^*(3)} \geq \dot{t}_{R_i^*(1)} + 3t > \dot{t}_{r_i'} + t$. As s^* is at $\dot{p}_{r_i'}$ either at time $t_{R_i^*(3)}$ or at time $\dot{t}_{R_i^*(3)}$, $\hat{t}_{r_i'}$ cannot be later than that time. Therefore, $|R_i^*| \leq 3$. (If $i = k$, observe again that *OPT* cannot accept any request that starts after $\hat{t}_{r_k'}$.) Thus, we have $P_{R_i^*} \leq 2r + r - c = 3r - c$.

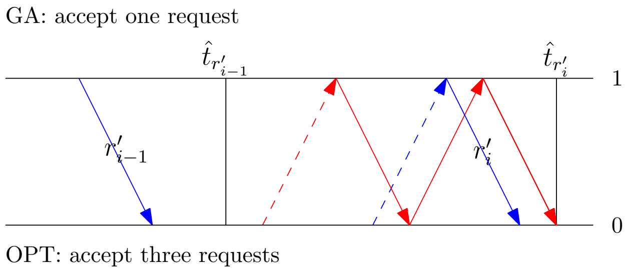


Figure 8: Illustration of R_i^* and R_i' in Case 2.1.1

Case 2.1.2: *OPT* accepts $R_i^*(1)$ without cost. Note that $p_{R_i^*(1)} = \dot{p}_{r_{i-1}'}$. See Fig. 9 for an illustration. If $|R_i^*| \leq 2$, we have $P_{R_i^*} \leq 2r < 3r - c$ as required. Assume $|R_i^*| \geq 3$. Because any request with pick-up location $\dot{p}_{r_i'}$ that starts during period $[\dot{t}_{r_{i-1}'}, t_{r_i'} - t]$ could be accepted by GA, we have $t_{R_i^*(1)} > t_{r_i'} - t$. Thus $\dot{t}_{R_i^*(1)} + t > \dot{t}_{r_i'}$. *OPT* accepts at most one request after $\dot{t}_{R_i^*(1)}$ during period $[\hat{t}_{r_{i-1}'}, \hat{t}_{r_i'})$ as any further accepted requests will be assigned to R_{i+1}^*. Therefore, we have $P_{R_i^*} \leq 2r$. Thus, $P_{R_i^*}/P_{R_i'} \leq \frac{2r}{r-c} \leq \frac{3r-c}{r-c}$.

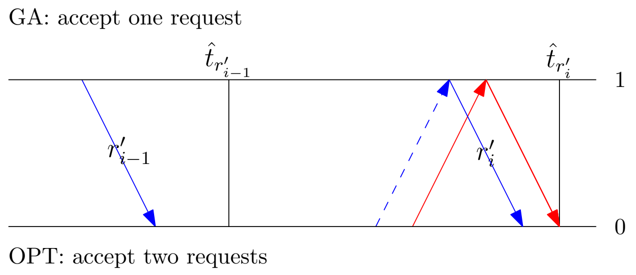


Figure 9: Illustration of R^*_i and R'_i in Case 2.1.2

Case 2.2: $\hat{t}_{r'_{i-1}} < \hat{t}_{r'_i}$ and $\hat{t}_{r'_{i-1}} \geq t_{r'_i}$. We show that $|R^*_i| \leq 2$ and hence $P_{R^*_i} \leq 2r$. Recall that s^* is at $\dot{p}_{r'_i}$ at time $\hat{t}_{r'_{i-1}}$.

Because $\hat{t}_{r'_{i-1}} \geq t_{r'_i}$, *OPT* can only accept one request with cost (and the end time of that request becomes $\hat{t}_{r'_i}$) or two requests without cost (and the end time of the second request becomes $\hat{t}_{r'_i}$) before time $\hat{t}_{r'_i}$. Hence $P_{R^*_i} \leq 2r$ and $P_{R^*_i}/P_{R'_i} \leq \frac{2r}{r-c} < \frac{3r-c}{r-c}$.

Case 2.3: $\hat{t}_{r'_{i-1}} \geq \hat{t}_{r'_i}$. The *OPT* period i is empty and, by definition, $R^*_i = \emptyset$. Therefore, $P^*_i = 0$ and $P_{R^*_i}/P_{R'_i} < \frac{3r-c}{r-c}$.

As we have shown that $P_{R^*_i}/P_{R'_i} \leq \frac{3r-c}{r-c}$ holds for all $1 \leq i \leq k$, we have $P_{R^*}/P_{R'} \leq \left(\sum_{i=1}^{k} \frac{3r-c}{r-c} P_{R'_i}\right) / \left(\sum_{i=1}^{k} P_{R'_i}\right) = \frac{3r-c}{r-c}$. □

Theorem 10. *Algorithm GA has competitive ratio at most $1 + 2\lceil\frac{b_u - b_l}{2t}\rceil$ for 1S2L-V if $c = r$. In particular, it is 3-competitive if $0 < b_u \leq t$.*

Proof. We prove the theorem by induction over the size of R'.

Base Case: $|R'| = 1$.

As GA only accepts requests without cost if $c = r$, we have $P_{R'} = r$. Before time $\tilde{t}_{r'_1}$, *OPT* can only accept requests which start during period $[0, \tilde{t}_{r'_1} + b_u)$ and have pick-up location 1 (note that s' is at point 0 and rejects all such requests). s' reaches location 1 at time $t_{r'_1} + t$ and could serve any request which starts no earlier than $t_{r'_1} + t$ and has pick-up location 1. Thus after time $\tilde{t}_{r'_1}$, *OPT* can only accept requests that start during period $[\tilde{t}_{r'_1} + b_l, \infty)$ and have pick-up location 0, and requests that start during period $[\tilde{t}_{r'_1} + b_l, t_{r'_1} + t)$ and have pick-up location 1.

We can assume without loss of generality that *OPT* only accepts requests without cost. As reasoned above, the earliest possible start time of a request with pick-up location 0 that *OPT* can accept is $\tilde{t}_{r'_1} + b_l$. The latest possible start time of a request with pick-up location 1 that OPT can accept is at most $\tilde{t}_{r'_1} + b_u$ if the request arrives before $\tilde{t}_{r'_1}$ and strictly smaller than $t_{r'_1} + t$ if the request arrives after $\tilde{t}_{r'_1}$.

First, consider the case that $\tilde{t}_{r'_1} + b_u \geq t_{r'_1} + t$. We bound the maximum number of pairs of requests (one with pick-up location 0 and the next with pick-up location 1) that OPT can accept. As the last request with pick-up location 1 that OPT can accept has start time at most $\tilde{t}_{r'_1} + b_u$, the start time of the first request of the last pair that OPT accepts is at most $\tilde{t}_{r'_1} + b_u - t$. As the start times of consecutive pairs differ by at least $2t$, the number of pairs is bounded by $1 + \lfloor((\tilde{t}_{r'_1} + b_u - t) - (\tilde{t}_{r'_1} + b_l))/2t\rfloor = 1 + \lfloor(b_u - b_l - t)/2t\rfloor$. If $b_u - b_l - t$ is a multiple of $2t$, this bound is equal to $1 + (b_u - b_l - t)/2t = \lceil(b_u - b_l)/2t\rceil$. Otherwise, the bound is equal to $\lceil(b_u - b_l - t)/2t\rceil \leq \lceil(b_u - b_l)/2t\rceil$. After the last pair, OPT can accept at most one more request with pick-up location 0. Therefore, $|R^*| \leq 1 + 2 \cdot \lceil\frac{b_u - b_l}{2t}\rceil$. As $P_{R'} = r$ and $P_{R^*} = |R^*| \cdot r$, we get $P_{R^*}/P_{R'} \leq 1 + 2\lceil\frac{b_u - b_l}{2t}\rceil$.

Now, consider the case that $\tilde{t}_{r'_1} + b_u < t_{r'_1} + t$. Again, we consider the maximum number of pairs of requests (one with pick-up location 0 and the next with pick-up location 1) that OPT can accept. As the last request with pick-up location 1 that OPT can accept must have start time strictly smaller than $t_{r'_1} + t \leq \tilde{t}_{r'_1} + b_u + t$, the start time of the first request of the last pair that OPT accepts is strictly smaller than $\tilde{t}_{r'_1} + b_u$. The start times of consecutive pairs differ by at least $2t$. If $b_u - b_l$ is a multiple of $2t$, the number of pairs is bounded by $((\tilde{t}_{r'_1} + b_u) - (\tilde{t}_{r'_1} + b_l))/2t = (b_u - b_l)/2t = \lceil(b_u - b_l)/2t\rceil$. If $b_u - b_l$ is not a multiple of $2t$, the number of pairs is bounded by $1 + \lfloor(b_u - b_l)/2t\rfloor = \lceil(b_u - b_l)/2t\rceil$. In any case, the number of pairs is at most $\lceil(b_u - b_l)/2t\rceil$. After the last pair, OPT can accept at most one more request with pick-up location 0. Therefore, $|R^*| \leq 1 + 2 \cdot \lceil\frac{b_u - b_l}{2t}\rceil$. As $P_{R'} = r$ and $P_{R^*} = |R^*| \cdot r$, we get $P_{R^*}/P_{R'} \leq 1 + 2\lceil\frac{b_u - b_l}{2t}\rceil$.

Induction Step: We assume that $\frac{P_{R^*}}{P_{R'}} \leq 1 + 2\lceil\frac{b_u - b_l}{2t}\rceil$ holds for all instances with $|R'| \leq i$ and show that then $\frac{P_{R^*}}{P_{R'}} \leq 1 + 2\lceil\frac{b_u - b_l}{2t}\rceil$ also holds for all instances with $|R'| = i + 1$.

Consider an instance of 1S2L-V given by a request sequence R where GA accepts $i + 1$ requests. As GA accepts requests in order of increasing arrival time by Observation 2, GA accepts i requests before time $\tilde{t}_{r'_{i+1}}$. Let $\bar{R}$ be the sub-instance of R that contains all requests in R except r'_{i+1} and all requests that are released after r'_{i+1} (i.e., released at time $\tilde{t}_{r'_{i+1}}$ and processed after r'_{i+1}, or released after time $\tilde{t}_{r'_{i+1}}$) and that GA could accept instead of r'_{i+1}. By the inductive hypothesis, OPT can achieve profit at most $i \cdot (1 + 2\lceil \frac{b_u - b_l}{2t} \rceil) \cdot r$ on the request sequence $\bar{R}$. The increase in profit that OPT can achieve on request sequence R compared to $\bar{R}$ must be due to requests accepted without cost that start in the interval $[\tilde{t}_{r'_{i+1}} + b_l, \infty)$ as all requests that start earlier were presented before time $\tilde{t}_{r'_{i+1}}$ and are thus contained in $\bar{R}$. Let Q be a largest set of requests in $R \setminus \bar{R}$ with start times in $[\tilde{t}_{r'_{i+1}} + b_l, \infty)$ that can be accepted without cost by OPT. Clearly, $P_{R^*} \le P_{\bar{R}^*} + r \cdot |Q|$, where $\bar{R}^*$ denotes the requests accepted by an optimal solution for the instance $\bar{R}$.

We claim that the first request in Q must have pick-up location $p_{r'_{i+1}}$ as otherwise that request would have to be contained in $\bar{R}$, a contradiction to Q being a subset of $R \setminus \bar{R}$. To see this, assume that the request with earliest start time after $\tilde{t}_{r'_{i+1}} + b_l$ in Q has pick-up location $\dot{p}_{r'_{i+1}}$. Denote that request by r_j. If r_j was presented before r'_{i+1}, it is clearly contained in $\bar{R}$. If r_j was presented after r'_{i+1}, it is also contained in $\bar{R}$ because GA cannot accept it instead of r'_{i+1} (as s' is at location $p_{r'_{i+1}}$ after serving r'_i and GA accepts requests in order of increasing start times by Observation 2).

Before time $\tilde{t}_{r'_{i+1}}$, OPT can accept requests with pick-up location $\dot{p}_{r'_{i+1}}$ that start no later than $\tilde{t}_{r'_{i+1}} + b_u$. After time $\tilde{t}_{r'_{i+1}}$, OPT can only accept requests with pick-up location $\dot{p}_{r'_{i+1}}$ that start strictly before $t_{r'_{i+1}} + t$, because s' arrives at $\dot{p}_{r'_{i+1}}$ at time $t_{r'_{i+1}} + t$ and could serve any request with pick-up location $\dot{p}_{r'_{i+1}}$ from that time onward.

First, consider the case that $\tilde{t}_{r'_{i+1}} + b_u \ge t_{r'_{i+1}} + t$. The last request that OPT can accept with pick-up location $\dot{p}_{r'_{i+1}}$ starts no later than $\tilde{t}_{r'_{i+1}} + b_u$. After that request, OPT can accept at most one more request with pick-up location $p_{r'_{i+1}}$. To bound the size of Q, we bound the maximum number of pairs of requests (one with pick-up location $p_{r'_{i+1}}$ and the next with pick-up location $\dot{p}_{r'_{i+1}}$) that OPT can accept. As the last request with pick-up location $\dot{p}_{r'_{i+1}}$ that OPT can accept has start time at most $\tilde{t}_{r'_{i+1}} + b_u$, the start time of the first request of the last pair that OPT accepts is at most $\tilde{t}_{r'_{i+1}} + b_u - t$. As the start times of consecutive pairs differ by at least $2t$, the number of pairs is bounded by $1 + \lfloor ((\tilde{t}_{r'_{i+1}} + b_u - t) - (\tilde{t}_{r'_{i+1}} + b_l))/2t \rfloor = 1 + \lfloor (b_u - b_l - t)/2t \rfloor$. If $b_u - b_l - t$ is a multiple of $2t$, this bound is equal to $1 + (b_u - b_l - t)/2t = \lceil (b_u - b_l)/2t \rceil$. Otherwise, the bound is equal to $\lceil (b_u - b_l - t)/2t \rceil \le \lceil (b_u - b_l)/2t \rceil$. After the last pair, OPT can accept at most one more request with pick-up location $p_{r'_{i+1}}$. Therefore, $|Q| \le 1 + 2 \cdot \lceil \frac{b_u - b_l}{2t} \rceil$.

Now, consider the case that $\tilde{t}_{r'_{i+1}} + b_u < t_{r'_{i+1}} + t$. Again, we consider the maximum number of pairs of requests (one with pick-up location $p_{r'_{i+1}}$ and the next with pick-up location $\dot{p}_{r'_{i+1}}$) that OPT can accept. As the last request with pick-up location $\dot{p}_{r'_{i+1}}$ that OPT can accept must have start time strictly smaller than $t_{r'_{i+1}} + t \leq \tilde{t}_{r'_{i+1}} + b_u + t$, the start time of the first request of the last pair that OPT accepts is strictly smaller than $\tilde{t}_{r'_{i+1}} + b_u$. The start times of consecutive pairs differ by at least $2t$. If $b_u - b_l$ is a multiple of $2t$, the number of pairs is bounded by $((\tilde{t}_{r'_{i+1}} + b_u) - (\tilde{t}_{r'_{i+1}} + b_l))/2t = (b_u - b_l)/2t = \lceil (b_u - b_l)/2t \rceil$. If $b_u - b_l$ is not a multiple of $2t$, the number of pairs is bounded by $1 + \lfloor (b_u - b_l)/2t \rfloor = \lceil (b_u - b_l)/2t \rceil$. In any case, the number of pairs is at most $\lceil (b_u - b_l)/2t \rceil$. After the last pair, OPT can accept at most one more request with pick-up location $p_{r'_{i+1}}$. Therefore, $|Q| \leq 1 + 2 \cdot \lceil \frac{b_u - b_l}{2t} \rceil$.

In either case, $|Q| \leq 1 + 2\lceil \frac{b_u - b_l}{2t} \rceil$. Thus, $P_{R^*} \leq P_{\bar{R}^*} + r \cdot |Q| \leq i(1 + 2\lceil \frac{b_u - b_l}{2t} \rceil)r + (1 + 2\lceil \frac{b_u - b_l}{2t} \rceil)r = (i+1)(1 + 2\lceil \frac{b_u - b_l}{2t} \rceil)r$. As $P_{R'} = (i+1)r$, we get $P_{R^*}/P_{R'} \leq 1 + 2\lceil \frac{b_u - b_l}{2t} \rceil$. □

4. Conclusion

We have studied an online problem with one server and two locations that is motivated by applications such as car sharing and taxi dispatching. In particular, we have analyzed the effects of different constraints on the booking time of requests on the competitive ratio that can be achieved. For all variants of booking time constraints and costs for empty server movements we have given matching lower and upper bounds on the competitive ratio. The upper bounds are all achieved by the same greedy algorithm (GA). Interestingly, the size of the booking horizon does not affect the competitive ratio if $0 \leq c < r$, but the competitive ratio increases as $b_u - b_l$ increases if $c = r$.

A number of directions for future work arise from this work. In particular, it would be interesting to extend our results to the case of more than one server and more than two locations. It would be interesting to determine how the constraints on the booking time affect the competitive ratio for the general car-sharing problem with k servers and m locations.

Another direction could be the design and analysis of randomized algorithms that achieve a better competitive ratio than deterministic algorithms for the online car-sharing problem.

In this paper, we only considered requests having different pick-up and drop-off locations. It would also be interesting to consider requests having the same pick-up and drop-off location, i.e., the server picks up the customer at a location and drops off the customer at the same location after a period of serving time. Another generalization is the consideration of the case where a request is a round-trip consisting of two opposite rides [15].

Acknowledgments.

This work was partially supported by the China Postdoctoral Science Foundation (Grant No. 2016M592811), the China Scholarship Council (Grant No. 201706280058), the funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement number 754462, and by the two EPSRC grants EP/S033483/1 and EP/T01461X/1.

References

[1] N. R. Kaushik, S. M. Figueira, S. A. Chiappari, Flexible time-windows for advance reservation scheduling, in: 14th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2006), IEEE Computer Society, 2006, pp. 218–225. doi:10.1109/MASCOTS.2006.25.

[2] G. Berbeglia, J. Cordeau, G. Laporte, Dynamic pickup and delivery problems, European Journal of Operational Research 202 (1) (2010) 8–15. doi:10.1016/j.ejor.2009.04.024.

[3] N. Ascheuer, S. O. Krumke, J. Rambau, Online dial-a-ride problems: Minimizing the completion time, in: H. Reichel, S. Tison (Eds.), 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2000), Vol. 1770 of LNCS, Springer, 2000, pp. 639–650. doi:10.1007/3-540-46541-3_53.

[4] A. Bjelde, Y. Disser, J. Hackfeld, C. Hansknecht, M. Lipmann, J. Meißner, K. Schewior, M. Schlöter, L. Stougie, Tight bounds for online TSP on the line, in: P. N. Klein (Ed.), 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017), SIAM, 2017, pp. 994–1005. doi:10.1137/1.9781611974782.63.

[5] S. O. Krumke, W. de Paepe, D. Poensgen, M. Lipmann, A. Marchetti-Spaccamela, L. Stougie, On minimizing the maximum flow time in

the online dial-a-ride problem, in: T. Erlebach, G. Persiano (Eds.), Third International Workshop on Approximation and Online Algorithms (WAOA 2005), Vol. 3879 of LNCS, Springer, 2006, pp. 258–269. doi:10.1007/11671411_20.

[6] M. Grötschel, D. Hauptmeier, S. O. Krumke, J. Rambau, Simulation studies for the online-dial-a-ride problem, Preprint SC 99-09, Konrad-Zuse-Zentrum für Informationstechnik Berlin (1999).

[7] F. Yi, L. Tian, On the online dial-a-ride problem with time-windows, in: N. Megiddo, Y. Xu, B. Zhu (Eds.), First International Conference on Algorithmic Applications in Management (AAIM 2005), Vol. 3521 of LNCS, Springer, 2005, pp. 85–94. doi:10.1007/11496199_11.

[8] A. Christman, W. Forcier, A. Poudel, From theory to practice: maximizing revenues for on-line dial-a-ride, J. Comb. Optim. 35 (2) (2018) 512–529. doi:10.1007/s10878-017-0188-z.

[9] K. Böhmová, Y. Disser, M. Mihalák, R. Šrámek, Scheduling transfers of resources over time: Towards car-sharing with flexible drop-offs, in: E. Kranakis, G. Navarro, E. Chávez (Eds.), 12th Latin American Symposium on Theoretical Informatics (LATIN 2016), Vol. 9644 of LNCS, Springer, 2016, pp. 220–234. doi:10.1007/978-3-662-49529-2_17.

[10] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, New Jersey, 1993.

[11] V. Bonifaci, L. Stougie, Online k-server routing problems, Theory Comput. Syst. 45 (3) (2009) 470–485. doi:10.1007/s00224-008-9103-4.

[12] G. Ausiello, V. Bonifaci, L. Laura, The online prize-collecting traveling salesman problem, Information Processing Letters 107 (6) (2008) 199–204.

[13] A. Borodin, R. El-Yaniv, Online computation and competitive analysis, Cambridge University Press, 1998.

[14] M. O. Ball, M. Queyranne, Toward robust revenue management: Competitive analysis of online booking, Oper. Res. 57 (4) (2009) 950–963. doi:10.1287/opre.1080.0654.

[15] K. Luo, Y. Xu, H. Liu, Online scheduling of car-sharing request pairs between two locations, Journal of Combinatorial Optimization (2020) 1–24.