Tight analysis of the lazy algorithm for open online dial-a-ride[*]

Júlia Baligács[1\[0000−0003−2654−149X\]], Yann Disser[1\[0000−0002−2085−0454\]], Farehe Soheil[1\[0000−0002−0504−8834\]], and David Weckbecker[1\[0000−0003−3381−058X\]]

TU Darmstadt, Germany
{baligacs|disser|soheil|weckbecker}@mathematik.tu-darmstadt.de

Abstract. In the open online dial-a-ride problem, a single server has to deliver transportation requests appearing over time in some metric space, subject to minimizing the completion time. We improve on the best known upper bounds on the competitive ratio on general metric spaces and on the half-line, for both the preemptive and non-preemptive version of the problem. We achieve this by revisiting the algorithm LAZY recently suggested in [WAOA, 2022] and giving an improved and tight analysis. More precisely, we show that it has competitive ratio 2.457 on general metric spaces and 2.366 on the half-line. This is the first upper bound that beats known lower bounds of 2.5 for schedule-based algorithms as well as the natural REPLAN algorithm.

Keywords: online algorithms · dial-a-ride · competitive analysis.

1 Introduction

In the open online dial-a-ride problem, we are given a metric space (M, d) and have control of a server that can move at unit speed. Over time, *requests* of the form $(a, b; t)$ arrive. Here, $a \in M$ is the *starting position* of the request, $b \in M$ is its *destination*, and $t \in \mathbb{R}_{\geq 0}$ is the *release time* of the request. We consider the online variant of the problem, meaning that the server does not get to know all requests at time 0, but rather at the respective release times. Our task is to control the server such that it serves all requests, i.e., we have to move the server to position a, load the request $(a, b; t)$ there after its release time t, and then move to position b where we unload the request. The objective is to minimize the *completion time*, i.e., the time when all requests are served.

We assume that the server always starts at time 0 in some fixed point, which we call the *origin* $O \in M$. The server has a *capacity* $c \in (\mathbb{N} \cup \{\infty\})$ and is not allowed to load more than c requests at the same time. Furthermore, we consider the *non-preemptive* version of the problem, that is, the server may not unload a request preemptively at a point that is not the request's destination. In the dial-a-ride problem, a distinction is made between the open and the closed variant. In the *closed* dial-a-ride problem, the server has to return to the origin

---

| metric space | | old bounds | | new bounds |
		lower	upper	upper
general	non-preemptive	2.05	**2.618** [5]	**2.457** (Thm 1)
	preemptive	2.04	2.618	2.457
line	non-preemptive	**2.05** [10]	2.618	2.457
	preemptive	**2.04** [11]	**2.41** [11]	—
half-line	non-preemptive	**1.9** [25]	2.618	**2.366** (Thm 2)
	preemptive	**1.62** [25]	2.41	2.366

Table 1. State of the art of the open online dial-a-ride problem and overview of our results: Bold bounds are original results, other bounds are inherited.

after serving all requests. By contrast, in the *open* dial-a-ride problem, the server may finish anywhere in the metric space. In this work, we only consider the open variant of the problem. By letting $a = b$ for all requests $(a, b; t)$, we obtain the *online travelling salesperson problem (TSP)* as a special case of the dial-a-ride problem.

In this work, we only consider deterministic algorithms for the online dial-a-ride problem. As usual in competitive analysis, we measure the quality of a deterministic algorithm by comparing it to an optimum offline algorithm. The measure we apply is the completion time of a solution. For a given sequence of requests σ and an algorithm ALG, we denote by $\text{ALG}(\sigma)$ the completion time of the algorithm for request sequence σ. Analogously, we denote by $\text{OPT}(\sigma)$ the completion time of an optimal offline algorithm. For some $\rho \geq 1$, we say that an algorithm ALG is ρ-competitive if, for all request sequences σ, we have $\text{ALG}(\sigma) \leq \rho \cdot \text{OPT}(\sigma)$. The *competitive ratio* of ALG is defined as $\inf\{\rho \geq 1 \mid \text{ALG is } \rho\text{-competitive}\}$. The *competitive ratio of a problem* is defined as $\inf\{\rho \geq 1 \mid \text{there is some } \rho\text{-competitive algorithm}\}$

Our results. We consider the parametrized algorithm $\text{LAZY}(\alpha)$ that was presented in [5] and prove the following results (see Table table 1).

Our main result is an improved general upper bound for the open online dial-a-ride problem.

Theorem 1. *For* $\alpha = \frac{1}{2} + \sqrt{11/12}$, $\text{LAZY}(\alpha)$ *has a competitive ratio of* $\alpha + 1 \approx 2.457$ *for open online dial-a-ride on general metric spaces for every capacity* $c \in \mathbb{N} \cup \{\infty\}$.

Prior to our work, the best known general upper bound of $\varphi + 1 \approx 2.618$ on the competitive ratio for the open online dial-a-ride problem was achieved by $\text{LAZY}(\varphi)$ and it was shown that $\text{LAZY}(\alpha)$ has competitive ratio at least $\frac{3}{2} + \sqrt{11/12} \approx 2.457$ for any choice of α, even on the line [5]. This means that we give a conclusive analysis of $\text{LAZY}(\alpha)$ by achieving an improved upper bound that tightly matches the previously known lower bound. In particular, $\alpha = 1/2 + \sqrt{11/12}$ is the (unique) best possible waiting parameter for $\text{LAZY}(\alpha)$, even on the line. The best known general lower bound remains 2.05 [10].

Crucially, our upper bound beats, for the first time, a known lower bound of 2.5 for the class of *schedule-based* algorithms [8], i.e., algorithms that divide the execution into subschedules that are never interrupted. Historically, all upper bounds, prior to those via LAZY, were based on schedule-based algorithms [9,10]. Our result means that online algorithms cannot afford to irrevocably commit to serving some subset of requests if they hope to attain the best possible competitive ratio.

Secondly, our upper bound also beats the same lower bound of 2.5 for the REOPT (or REPLAN) algorithm [3], which simply reoptimizes its solution whenever new requests appear. While this algorithm is very natural and may be the first algorithm studied for the online dial-a-ride problem, it has eluded tight analysis up to this day. So far, it has been a canonical candidate for a best-possible algorithm. We finally rule it out.

In addition to the general bound above, we analyze LAZY(α) for open online dial-a-ride on the half-line, i.e., where $M = \mathbb{R}_{\geq 0}$, and show that, in this metric space, even better bounds on the competitive ratio are possible for different values of α. More precisely, we show the following.

Theorem 2. *For $\alpha = \frac{1+\sqrt{3}}{2}$, LAZY($\alpha$) has a competitive ratio of $\alpha + 1 \approx 2.366$ for open online dial-a-ride on the half-line for every capacity $c \in \mathbb{N} \cup \{\infty\}$.*

This further improves on the previous best known upper bound of 2.618 [5]. The best known lower bound is 1.9 [25].

We go on to show that the bound in Theorem 2 is best-possible for LAZY(α) over all parameter choices $\alpha \geq 0$.

Theorem 3. *For all $\alpha \geq 0$, LAZY(α) has a competitive ratio of at least $\alpha + 1 \approx 2.366$ for open online dial-a-ride on the half-line for every capacity $c \in \mathbb{N} \cup \{\infty\}$.*

In the preemptive version of the online dial-a-ride problem, the server is allowed to unload requests anywhere and pick them up later again. In this version, prior to our work, the best known upper bound on general metric spaces was 2.618 [5] and the best known upper bound on the line and the half-line was 2.41 [11]. Obviously, every non-preemptive algorithm can also be applied in the preemptive setting, however, its competitive ratio may degrade since the optimum might have to use preemption. Our algorithm LAZY repeatedly executes optimal solutions for subsets of requests and can be turned preemptive by using preemptive solutions. With this change, our analysis of LAZY still carries through in the preemptive case and improves the state of the art for general metric spaces and the half-line, but not the line. The best known lower bound in the preemptive version on general metric spaces is 2.04 [11] and the best known lower bound on the half-line is 1.62 [25].

Corollary 1. *The competitive ratio of the open preemptive online dial-a-ride problem with any capacity $c \in \mathbb{N} \cup \{\infty\}$ is upper bounded by*

a) $\frac{3}{2} + \sqrt{\frac{11}{12}} \approx 2.457$ and this bound is achieved by LAZY $\left(\frac{1}{2} + \sqrt{\frac{11}{12}}\right)$,

b) $1 + \frac{1+\sqrt{3}}{2} \approx 2.366$ on the half-line and this bound is achieved by LAZY($\frac{1+\sqrt{3}}{2}$).

Related work. Two of the most natural algorithms for the online dial-a-ride problem are IGNORE and REPLAN. The basic idea of IGNORE is to repeatedly follow an optimum schedule over the currently unserved requests and ignoring all requests released during its execution. The competitive ratio of this algorithm is known to be exactly 4 [8,21]. By contast, the main idea of REPLAN is to start a new schedule over all unserved requests whenever a new request is released. While this algorithm has turned out to be notoriously difficult to analyze, it is known that its competitive ratio is at least 2.5 [3] and at most 4 [8]. Several variants of these algorithms have been proposed such as SMARTSTART [21], SMARTERSTART [10] or WAITORIGNORE [25], which lead to improvements on the best known bounds on the competitive ratio of the dial-a-ride problem. In this work, we study the recently suggested algorithm LAZY [5], which is known to achieve a competitive ratio of $\varphi + 1$ for the open online dial-a-ride problem, where $\varphi = \frac{\sqrt{5}+1}{2} \approx 1.618$ denotes the golden ratio.

For the preemptive version of the open dial-a-ride problem, the best known upper bound on general metric spaces is $\varphi + 1 \approx 2.618$ [5]. Bjelde et al. [11] proved a stronger upper bound of $1 + \sqrt{2} \approx 2.41$ for when the metric space is the line.

In terms of lower bounds, Birx et al. [10] were able to prove that every algorithm for the open online dial-a-ride problem has a competitive ratio of at least 2.05, even if the metric space is the line. This separates dial-a-ride from online TSP on the line, where it is known that the competitive ratio is exactly 2.04 [11]. For open online dial-a-ride on the half-line, Lipmann [25] established a lower bound of 1.9 for the non-preemptive version and a lower bound of 1.62 for the preemptive version.

For the closed variant of the online dial-a-ride problem, the competitive ratio is known to be exactly 2 on general metric spaces [1,3,14] and between 1.76 and 2 on the line [8,11]. On the half-line the best known lower bound is 1.71 [1] and the best known upper bound is 2 [1,14]. The TSP variant of the closed dial-a-ride problem is tightly analyzed with a competitive ratio of 2 on general metric spaces [1,3,14], of 1.64 on the line [3,11], and of 1.5 on the half-line [12].

Other variants of the problem have been studied in the literature. This includes settings where the request sequence has to fulfill some reasonable additional properties [12,16,22], where the server is presented with additional [4] or less [26] information, where the server has some additional abilities [13,19], where the server has to handle requests in a given order [15,19], or where we consider different objectives than the completion time [2,6,7,16,17,18,22,23,24]. Other examples include the study of randomized algorithms [21], or other metric spaces, such as a circle [20]. Moreover, it has been studied whether some natural classes of algorithms can have good competitive ratios. For example, *zealous* algorithms always have to move towards an unserved request or the origin [12]. A *schedule-based* algorithm operates in schedules that are not allowed to be interrupted. Birx [8] showed that all such algorithms have a competitive ratio of at least 2.5. Together with our results, this implies that schedule-based algorithms algorithms cannot be best-possible.

2   Algorithm description and notation

In this section, we define the algorithm Lazy introduced in [5]. The rough idea of the algorithm is to wait until several requests are revealed and then start a schedule serving them. Whenever a new request arrives, we check whether we can deliver all currently loaded requests and return to the origin in a reasonable time. If this is possible, we do so and begin a new schedule including the new requests starting from the origin. If this is not possible, we keep following the current schedule and consider the new request later.

More formally, a *schedule* is a sequence of actions specifying the server's behaviour, including its movement and where requests are loaded or unloaded. By $\text{OPT}[t]$, we denote an optimal schedule beginning in O at time 0 and serving all requests that are released not later than time t. By $\text{OPT}(t)$, we denote its completion time. Given a set of requests R and some point $x \in M$, we denote by $S(R, x)$ a shortest schedule serving all requests in R beginning from point x at some time after all requests in R are released. In other words, we can ignore the release times of the requests when computing $S(R, x)$. As waiting is not beneficial for the server if there are no release times, the *length of the schedule*, i.e., the distance the server travels, is the same as the time needed to complete it and we denote this by $|S(R, x)|$.

Now that we have established the notation needed, we can describe the algorithm (cf. Algorithm 1). By t, we denote the current time. By p_t, we denote the position of the server at time t, and by R_t, we denote the set of requests that have been released but not served until time t. The variable i is a counter over the schedules started by the algorithm. The waiting parameter $\alpha \geq 1$ specifies how long we wait before starting a schedule. The algorithm uses the following commands: DELIVER_AND_RETURN orders the server to finish serving all currently loaded requests and return to O in the fastest possible way, WAIT_UNTIL(t) orders the server to remain at its current location until time t, and FOLLOW_SCHEDULE(S) orders the server to execute the actions defined by schedule S. When any of these commands is invoked, the server aborts what it is doing and executes the new command. Whenever the server has completed a command, we say that it becomes *idle*.

We make a few comments for illustration of the algorithm. If the server returns to the origin upon receiving a request, we say that the schedule it was currently following is *interrupted*. Observe that, due to interruption, the sets $R^{(i)}$ are not necessarily disjoint. Also, observe that $p^{(1)} = O$, and if schedule $S^{(i)}$ was interrupted, we have $p^{(i+1)} = O$ and $t^{(i+1)} = \alpha \cdot \text{OPT}(t^{(i+1)})$. If $S^{(i)}$ was not interrupted, $p^{(i+1)}$ is the ending position of $S^{(i)}$.

The following observations were already noted in [5] and follow directly from the definitions above and the fact that requests in $R^{(i)} \setminus R^{(i-1)}$ were released after time $t^{(i-1)}$.

Observation 1 ([5]). *For every request sequence, the following hold.*

a) For every $i > 1$, $\text{OPT}(t^{(i)}) \geq t^{(i-1)} \geq \alpha \cdot \text{OPT}(t^{(i-1)})$.

Algorithm 1: LAZY(α)

---
initialize: $i \leftarrow 0$

---
upon receiving a request:
if *server can serve all loaded requests and return to O until time $\alpha \cdot \mathrm{OPT}(t)$*
then
$\quad\lfloor$ **execute** DELIVER_AND_RETURN

---
upon becoming idle:
if $t < \alpha \cdot \mathrm{OPT}(t)$ **then**
$\quad\mid$ **execute** WAIT_UNTIL($\alpha \cdot \mathrm{OPT}(t)$)
else if $R_t \neq \emptyset$ **then**
$\quad\mid$ $i \leftarrow i+1,\ R^{(i)} \leftarrow R_t,\ t^{(i)} \leftarrow t,\ p^{(i)} \leftarrow p_t$
$\quad\mid$ $S^{(i)} \leftarrow S(R^{(i)}, p^{(i)})$
$\quad\lfloor$ **execute** FOLLOW_SCHEDULE($S^{(i)}$)

---

b) For every $x, y \in M$ and every subset of requests R, we have $|S(R,x)| \leq d(x,y) + |S(R,y)|$.

c) Let $i > 1$ and assume that $S^{(i-1)}$ was not interrupted. Let a be the starting position of the request in $R^{(i)}$ that is picked up first by $\mathrm{OPT}(t^{(i)})$. Then,

$$\mathrm{OPT}(t^{(i)}) \geq t^{(i-1)} + |S(R^{(i)}, a)| \geq \alpha \cdot \mathrm{OPT}(t^{(i-1)}) + |S(R^{(i)}, a)|.$$

3   Factor-revealing approach

The results in this paper were informed by a factor-revealing technique, inspired by a similar approach of Bienkowski et al. [6], to analyze a specific algorithm ALG, in our case LAZY. The technique is based on a formulation of the adversary problem, i.e., the problem of finding an instance that maximizes the competitive ratio, as an optimization problem of the form

$$\max\left\{ \frac{\mathrm{ALG}(x)}{\mathrm{OPT}(x)} \;\middle|\; x \text{ describes a dial-a-ride instance} \right\}. \tag{1}$$

An optimum solution to this problem immediately yields the competitive ratio of ALG. Of course, we cannot hope to solve this optimization problem or even describe it with a finite number of variables. The factor-revealing approach consists in relaxing (1) to a practically solvable problem over a finite number of variables.

The key is to select a set of variables that captures the structure of the problem well enough to allow for meaningful bounds. In our case, we can, for example, introduce variables for the starting position and duration of the second-to-last as well as for the last schedule. We then need to relate those variables via constraints that ensure that an optimum solution to the relaxed problem actually has a realization as a dial-a-ride instance. For example, we might add the constraint that the distance between the starting positions of the last two schedules is upper bounded by the duration of the second-to-last schedule.

The power of the factor-revealing approach is that it allows to follow an iterative process for deriving structurally crucial inequalities: When solving the relaxed optimization problem, we generally have to expect an optimum solution that is not realizable and overestimates the competitive ratio. We can then focus our efforts on understanding why the corresponding variable assignment cannot be realized by a dial-a-ride instance. Then, we can introduce additional variables and constraints to exclude such solutions. In this way, the unrealizable solutions inform our analysis in the sense that we obtain bounds on the competitive ratio that can be proven analytically by only using the current set of variables and inequalities. Once we obtain a realizable lower bound, we thus have found the exact competitive ratio of the algorithm under investigation.

In order to practically solve the relaxed optimization problems, we limit ourselves to linear programs (LPs). Note that the objective of (1) is linear if we normalize to $\text{OPT}(x) = 1$. We can do this, since the competitive ratio is invariant with respect to rescaling the metric space and release times of requests. Another advantage of using linear programs is that we immediately obtain a formal proof of the optimum solution from an optimum solution to the LP dual. Of course, the correctness of the involved inequalities still needs to be established.

In the remainder of this paper, we present a purely analytic proof of our results. Many of the inequalities we derive in lemmas were informed by a factor-revealing approach via a linear program with a small number of binary variables. This means that we additionally need to branch on all binary variables in order to obtain a formal proof via LP duality. We refer to Appendix A for more details of the binary program that informed our results for the half-line.

4 Analysis on general metric spaces

This section is concerned with the proof of Theorem 1. For the remainder of this section, let $(r_1, \ldots, r_n)$ be some fixed request sequence. Let k be the number of schedules started by $\text{LAZY}(\alpha)$, and let $S^{(i)}$, $t^{(i)}$, $p^{(i)}$, $R^{(i)}$ $(1 \le i \le k)$ be defined as in the algorithm. Note that we slightly abuse notation here because k, $S^{(i)}$, $t^{(i)}$, $p^{(i)}$, and $R^{(i)}$ depend on α. As it will always be clear from the context what α is, we allow this implicit dependency in the notation.

As it will be crucial for the proof in which order OPT and LAZY serve requests, we introduce the following notation. Let

- $r_{f,\text{OPT}}^{(i)} = (a_{f,\text{OPT}}^{(i)}, b_{f,\text{OPT}}^{(i)}; t_{f,\text{OPT}}^{(i)})$ be the first request in $R^{(i)}$ picked up by $\text{OPT}[t^{(i)}]$,
- $r_{l,\text{OPT}}^{(i)} = (a_{l,\text{OPT}}^{(i)}, b_{l,\text{OPT}}^{(i)}; t_{l,\text{OPT}}^{(i)})$ be the last request in $R^{(i)}$ delivered by $\text{OPT}[t^{(i+1)}]$,
- $r_{f,\text{LAZY}}^{(i)} = (a_{f,\text{LAZY}}^{(i)}, b_{f,\text{LAZY}}^{(i)}; t_{f,\text{LAZY}}^{(i)})$ be the first request in $R^{(i)}$ picked up by $\text{LAZY}(\alpha)$,
- $r_{l,\text{LAZY}}^{(i)} = (a_{l,\text{LAZY}}^{(i)}, p^{(i+1)}; t_{l,\text{LAZY}}^{(i)})$ be the last request in $R^{(i)}$ delivered by $\text{LAZY}(\alpha)$.

Definition 1. *We say that the i-th schedule is α-good if*

a) $|S^{(i)}| \leq \text{OPT}(t^{(i)})$ *and*
b) $t^{(i)} + |S^{(i)}| \leq (1 + \alpha) \cdot \text{OPT}(t^{(i)})$.

In this section, we prove by induction on i that, for $\alpha \geq \frac{1}{2} + \sqrt{11/12}$, every schedule is α-good. Note that this immediately implies Theorem 1.

As our work builds on [5], the first few steps of our proof are the same as in [5]. For better understandability and reading flow, we repeat the proofs of some important but simple steps and mark the results with appropriate citations. The results starting with Lemma 2 are new and improve on the analysis in [5].

We begin with proving the base case.

Observation 2 (Base case, [5]). *For every $\alpha \geq 1$, the first schedule is α-good.*

Proof. Recall that $S^{(1)}$ begins in O and is the shortest tour serving all requests in $R^{(1)}$. $\text{OPT}[t^{(1)}]$ begins in O and serves all requests in $R^{(1)}$, too, which yields $|S^{(1)}| \leq \text{OPT}(t^{(1)})$. The fact that we have $t^{(1)} = \alpha \cdot \text{OPT}(t^{(1)})$ implies $t^{(1)} + |S^{(1)}| \leq (1 + \alpha) \cdot \text{OPT}(t^{(1)})$. $\square$

Next, we observe briefly that the induction step is not too difficult when the last schedule was interrupted.

Observation 3 (Interruption case, [5]). *Let $\alpha \geq 1$. Assume that schedule $S^{(i)}$ was interrupted. Then, $S^{(i+1)}$ is α-good.*

Proof. If schedule $S^{(i)}$ was interrupted, we have $p^{(i+1)} = O$ and $t^{(i+1)} = \alpha \cdot \text{OPT}(t^{(i+1)})$. Therefore, $|S^{(i+1)}| = |S(R^{(i+1)}, O)| \leq \text{OPT}(t^{(i+1)})$ and $t^{(i+1)} + |S^{(i+1)}| \leq (1 + \alpha) \cdot \text{OPT}(t^{(i+1)})$. $\square$

For this reason, we will assume in many of the following statements that the schedule $S^{(i)}$ was not interrupted.

By careful observation of the proof in [5], one can see that the following fact already holds for smaller α. For convenience, we repeat the proof of the following Lemma with an adapted value of α.

Lemma 1 ([5]). *Let $\alpha \geq \frac{1+\sqrt{17}}{4} \approx 1.281$ and $i \in \{1, \ldots, k-1\}$. If $S^{(i)}$ is α-good, then $|S^{(i+1)}| \leq \text{OPT}(t^{(i+1)})$.*

Proof. First, observe that if $S^{(i)}$ was interrupted, we have $p^{(i+1)} = O$. Note that $\text{OPT}(t^{(i+1)})$ begins in O and serves all requests in $R^{(i+1)}$ so that we have

$$|S^{(i+1)}| = |S(R^{(i+1)}, O)| \leq \text{OPT}(t^{(i+1)}).$$

Therefore, assume from now on that $S^{(i)}$ was not interrupted. Also, if $\text{OPT}[t^{(i+1)}]$ serves $r_{l,\text{LAZY}}^{(i)}$ at $p^{(i+1)}$ before collecting any request from $R^{(i+1)}$, we trivially have

$$|S^{(i+1)}| = |S(R^{(i+1)}, p^{(i+1)}| \leq \text{OPT}(t^{(i+1)}).$$

Therefore, assume additionally that $\text{OPT}[t^{(i+1)}]$ collects $r_{f,\text{OPT}}^{(i+1)}$ before serving $r_{l,\text{LAZY}}^{(i)}$. Next, we prove the following assertion.

Claim: In the setting described above, we have

$$d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}) \leq \left(1 + \frac{2}{\alpha} - \alpha\right) \text{OPT}(t^{(i)}). \tag{2}$$

To prove the claim, note that $r_{f,\text{OPT}}^{(i+1)}$ is released not earlier than $\alpha \cdot \text{OPT}(t^{(i)})$. Since we assume that $\text{OPT}(t^{(i+1)})$ collects $r_{f,\text{OPT}}^{(i+1)}$ before serving $r_{l,\text{LAZY}}^{(i)}$ at $p^{(i+1)}$, we obtain

$$\text{OPT}(t^{(i+1)}) \geq \alpha \cdot \text{OPT}(t^{(i)}) + d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}). \tag{3}$$

Upon the arrival of the last request in $R^{(i)}$, we have $\text{OPT}(t) = \text{OPT}(t^{(i+1)})$ and the server can finish its current schedule and return to the origin in time $t^{(i)} + |S^{(i)}| + d(p^{(i+1)}, O)$. As we assume that $S^{(i)}$ was not interrupted, this yields

$$t^{(i)} + |S^{(i)}| + d(p^{(i+1)}, O) > \alpha \cdot \text{OPT}(t^{(i+1)}). \tag{4}$$

Combined, we obtain that

$$
\begin{aligned}
d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}) &\overset{(3)}{\leq} \text{OPT}(t^{(i+1)}) - \alpha \cdot \text{OPT}(t^{(i)}) \\
&\overset{(4)}{\leq} \frac{1}{\alpha} \cdot \left(t^{(i)} + |S^{(i)}| + d(p^{(i+1)}, O)\right) - \alpha \cdot \text{OPT}(t^{(i)}) \\
&\overset{S^{(i)} \alpha\text{-good}}{\leq} \frac{1}{\alpha} \cdot \left((1+\alpha) \cdot \text{OPT}(t^{(i)}) + d(p^{(i+1)}, O)\right) - \alpha \cdot \text{OPT}(t^{(i)}) \\
&\leq \left(1 + \frac{2}{\alpha} - \alpha\right) \text{OPT}(t^{(i)}),
\end{aligned}
$$

where we have used in the last inequality that $d(p^{(i+1)}, O) \leq \text{OPT}(t^{(i)})$ because $\text{OPT}(t^{(i)})$ begins in O and has to serve $r_{l,\text{LAZY}}^{(i)}$ at $p^{(i+1)}$. This completes the proof of the claim.

Now, we turn back to proving Lemma 1. We obtain

$$
\begin{aligned}
|S^{(i+1)}| &\leq d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
&\overset{\text{Obs 1c)}}{\leq} d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + \text{OPT}(t^{(i+1)}) - \alpha \cdot \text{OPT}(t^{(i)}) \\
&\overset{(2)}{\leq} \left(1 + \frac{2}{\alpha} - 2\alpha\right) \text{OPT}(t^{(i)}) + \text{OPT}(t^{(i+1)}) \\
&\leq \text{OPT}(t^{(i+1)}),
\end{aligned}
$$

where the last inequality follows from the fact that $1 + \frac{2}{\alpha} - 2\alpha \leq 0$ if and only if $\alpha \geq \frac{1+\sqrt{17}}{4} \approx 1.2808$. $\square$

Recall that the goal of this section is to prove that every schedule is α-good. So far, we have proven the base case (cf. Observation 2) and

$|S^{(i+1)}| \leq \text{OPT}(t^{(i+1)})$ (Lemma 1) in the induction step. It remains to show that $t^{(i+1)} + |S^{(i+1)}| \leq (1+\alpha) \cdot \text{OPT}(t^{(i+1)})$ assuming $S^{(1)}, \ldots, S^{(i)}$ are α-good. In Observation 3, we have already seen that this holds if $S^{(i)}$ was interrupted. To show that the induction step also holds if $S^{(i)}$ was not interrupted, we distinguish several cases for the order in which OPT serves the requests. We begin with the case that $\text{OPT}[t^{(i+1)}]$ picks up some request in $R^{(i+1)}$ before serving $r_{l,\text{LAZY}}^{(i)}$, i.e., that $\text{OPT}[t^{(i+1)}]$ does not follow the order of the $S^{(i)}$.

Lemma 2. *Let $\alpha \geq 1$. Assume that $S^{(i)}$ is α-good and was not interrupted, and that $\text{OPT}[t^{(i+1)}]$ picks up $r_{f,\text{OPT}}^{(i+1)}$ before serving $r_{l,\text{LAZY}}^{(i)}$. Then, $t^{(i+1)} + |S^{(i+1)}| \leq (1+\alpha) \cdot \text{OPT}(t^{(i+1)})$.*

Proof. Using the order in which OPT handles the requests, we obtain the following. After picking up $r_{f,\text{OPT}}^{(i+1)}$ at $a_{f,\text{OPT}}^{(i+1)}$ after time $t^{(i)}$, $\text{OPT}[t^{(i+1)}]$ has to serve $r_{l,\text{LAZY}}^{(i)}$ at $p^{(i+1)}$ so that

$$\text{OPT}(t^{(i+1)}) \geq t^{(i)} + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}). \tag{5}$$

After finishing schedule $S^{(i)}$, the server either waits until time $\alpha \cdot \text{OPT}(t^{(i+1)})$ or immediately starts the next schedule, i.e., we have

$$t^{(i+1)} = \max\{\alpha \cdot \text{OPT}(t^{(i+1)}), t^{(i)} + |S^{(i)}|\}.$$

If $t^{(i+1)} = \alpha \cdot \text{OPT}(t^{(i+1)})$, the assertion follows immediately from Lemma 1. Thus, assume $t^{(i+1)} = t^{(i)} + |S^{(i)}|$. This yields

$$
\begin{aligned}
t^{(i+1)} + |S^{(i+1)}| \; &\overset{\text{Obs 1b)}}{\leq} \; t^{(i)} + |S^{(i)}| + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
&\overset{S^{(i)} \, \alpha\text{-good}}{\leq} \; (1+\alpha) \cdot \text{OPT}(t^{(i)}) + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \\
&\qquad\qquad + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
&\overset{\text{Obs 1a)}}{\leq} \; \frac{1+\alpha}{\alpha} t^{(i)} + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
&\overset{\text{Obs 1c)}}{\leq} \; \frac{1}{\alpha} t^{(i)} + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + \text{OPT}(t^{(i+1)}) \\
&\overset{(5)}{\leq} \; \frac{1}{\alpha} \left(\text{OPT}(t^{(i+1)}) - d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \right) + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \\
&\qquad\qquad + \text{OPT}(t^{(i+1)}) \\
&= \left(1 + \frac{1}{\alpha} \right) \text{OPT}(t^{(i+1)}) + \left(1 - \frac{1}{\alpha} \right) d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \\
&\leq 2 \cdot \text{OPT}(t^{(i+1)}),
\end{aligned}
$$

where we have used in the last inequality that $d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \leq \text{OPT}(t^{(i+1)})$ as $\text{OPT}[t^{(i+1)}]$ has to visit both points. $\qquad\square$

Next, we consider the case where OPT handles $r_{l,\text{LAZY}}^{(i)}$ and $r_{f,\text{OPT}}^{(i+1)}$ in the same order as LAZY.

Lemma 3. *Let $\alpha \geq 1$. Assume that schedules $S^{(1)}, \ldots, S^{(i)}$ are α-good, $S^{(i)}$ was not interrupted, and $\text{OPT}[t^{(i+1)}]$ serves $r_{l,\text{LAZY}}^{(i)}$ before collecting $r_{f,\text{OPT}}^{(i+1)}$. If we have $d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + \text{OPT}(t^{(i)}) \leq \alpha \cdot \text{OPT}(t^{(i+1)})$, then $t^{(i+1)} + |S^{(i+1)}| \leq (1 + \alpha) \cdot \text{OPT}(t^{(i+1)})$.*

Proof. Similarly as in the proof of Lemma 2, we can assume

$$t^{(i+1)} = t^{(i)} + |S^{(i)}|. \tag{6}$$

We have

$$
\begin{aligned}
t^{(i+1)} + |S^{(i+1)}| &\overset{\text{Obs 1b)}}{\leq} t^{(i)} + |S^{(i)}| + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
&\overset{S^{(i)} \alpha\text{-good}}{\leq} (1 + \alpha) \cdot \text{OPT}(t^{(i)}) + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \\
&\qquad + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
&\overset{\text{Obs 1c)}}{\leq} (1 + \alpha) \cdot \text{OPT}(t^{(i)}) + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \\
&\qquad + \text{OPT}(t^{(i+1)}) - \alpha \cdot \text{OPT}(t^{(i)}) \\
&= \text{OPT}(t^{(i+1)}) + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + \text{OPT}(t^{(i)}) \\
&\leq (1 + \alpha) \cdot \text{OPT}(t^{(i+1)}),
\end{aligned}
$$

where the last inequality follows from the assumption that $d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + \text{OPT}(t^{(i)}) \leq \alpha \cdot \text{OPT}(t^{(i+1)})$. □

Now that we have proven the case described in Lemma 3, we will assume in the following that

$$d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) > \alpha \cdot \text{OPT}(t^{(i+1)}) - \text{OPT}(t^{(i)}). \tag{7}$$

The following lemma states that, in this case, the $(i-1)$-th schedule (if it exists) was interrupted, i.e., the i-th schedule starts in the origin at time $\alpha \cdot \text{OPT}(t^{(i)})$.

Lemma 4. *Let $\alpha \geq \frac{1+\sqrt{3}}{2} \approx 1.366$. Assume that the i-th schedule is α-good and was not interrupted, and $\text{OPT}[t^{(i+1)}]$ serves $r_{l,\text{LAZY}}^{(i)}$ before collecting $r_{f,\text{OPT}}^{(i+1)}$. If (7) holds, then $p^{(i)} = O$ and $t^{(i)} = \alpha \cdot \text{OPT}(t^{(i)})$.*

Proof. If $i = 1$, we obviously have $p^{(i)} = O$ and $t^{(i)} = \alpha \cdot \text{OPT}(t^{(i)})$. Thus, assume that $i \geq 2$. If $r_{l,\text{LAZY}}^{(i)} \in (R^{(i-1)} \cap R^{(i)})$, schedule $S^{(i-1)}$ was interrupted and, thus, the statement holds. Otherwise, request $r_{l,\text{LAZY}}^{(i)}$ is released while schedule $S^{(i-1)}$ is running, i.e., $t_{l,\text{LAZY}}^{(i)} \geq t^{(i-1)} \geq \alpha \cdot \text{OPT}(t^{(i-1)})$. Combining this with the assumption that $\text{OPT}[t^{(i+1)}]$ serves $r_{l,\text{LAZY}}^{(i)}$ before collecting $r_{f,\text{OPT}}^{(i+1)}$, we obtain

$$\text{OPT}(t^{(i+1)}) \geq t_{l,\text{LAZY}}^{(i)} + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) \geq \alpha \cdot \text{OPT}(t^{(i-1)}) + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}). \tag{8}$$

Rearranging yields

$$\alpha \cdot \mathrm{OPT}(t^{(i-1)}) \overset{(8)}{\leq} \mathrm{OPT}(t^{(i+1)}) - d(p^{(i+1)}, a_{f,\mathrm{OPT}}^{(i+1)})$$

$$\overset{(7)}{<} \mathrm{OPT}(t^{(i)}) - (\alpha - 1)\mathrm{OPT}(t^{(i+1)})$$

$$\overset{\mathrm{Obs\ 1a)}}{\leq} (1 + \alpha - \alpha^2)\mathrm{OPT}(t^{(i)}),$$

which is equivalent to

$$\mathrm{OPT}(t^{(i-1)}) < \left(1 + \frac{1}{\alpha} - \alpha\right)\mathrm{OPT}(t^{(i)}). \tag{9}$$

By the assumption that $S^{(i-1)}$ is α-good, the server finishes schedule $S^{(i-1)}$ not later than time $(\alpha + 1) \cdot \mathrm{OPT}(t^{(i-1)})$. Thus, at the time where request $r_{l,\mathrm{LAZY}}^{(i)}$ is released, the server can serve all loaded requests and return to the origin by time

$$\max\{(\alpha + 1)\mathrm{OPT}(t^{(i-1)}), t_{l,\mathrm{LAZY}}^{(i)}\} + \mathrm{OPT}(t^{(i-1)}).$$

We have

$$(\alpha + 2) \cdot \mathrm{OPT}(t^{(i-1)}) \overset{(9)}{<} (\alpha + 2)\left(1 + \frac{1}{\alpha} - \alpha\right)\mathrm{OPT}(t^{(i)})$$

$$= \left(3 + \frac{2}{\alpha} - \alpha^2 - \alpha\right)\mathrm{OPT}(t^{(i)})$$

$$\leq \alpha \cdot \mathrm{OPT}(t^{(i)}),$$

where the last inequality holds for $\alpha \geq 1.343$. Furthermore, as $r_{l,\mathrm{LAZY}}^{(i)} \in R^{(i)}$, it holds that

$$t_{l,\mathrm{LAZY}}^{(i)} + \mathrm{OPT}(t^{(i-1)}) \leq \mathrm{OPT}(t^{(i)}) + \mathrm{OPT}(t^{(i-1)})$$

$$\overset{(9)}{\leq} \left(2 + \frac{1}{\alpha} - \alpha\right)\mathrm{OPT}(t^{(i)}) \leq \alpha \cdot \mathrm{OPT}(t^{(i)}),$$

where the last inequality holds for $\alpha \geq \frac{1+\sqrt{3}}{2} \approx 1.366$. This implies that the server can return to the origin by time $\alpha \cdot \mathrm{OPT}(t^{(i)})$, i.e., we have $p^{(i)} = O$. □

We now come to the technically most involved case.

Lemma 5. *Let $\alpha \geq \frac{1}{2} + \sqrt{11/12} \approx 1.457$. Assume that the i-th schedule is α-good and was not interrupted, and $\mathrm{OPT}[t^{(i+1)}]$ serves $r_{l,\mathrm{LAZY}}^{(i)}$ before collecting $r_{f,\mathrm{OPT}}^{(i+1)}$. If (7) holds, then $t^{(i+1)} + |S^{(i+1)}| \leq (1 + \alpha) \cdot \mathrm{OPT}(t^{(i+1)})$.*

Proof. We begin by proving the following assertion.
Claim: $\mathrm{OPT}[t^{(i+1)}]$ serves all requests in $R^{(i)}$ before picking up $r_{f,\mathrm{OPT}}^{(i+1)}$ in $a_{f,\mathrm{OPT}}^{(i+1)}$.
To prove the claim, assume otherwise, i.e., that $\mathrm{OPT}[t^{(i+1)}]$ serves $r_{l,\mathrm{OPT}}^{(i)}$ after

collecting $r_{f,\mathrm{OPT}}^{(i+1)}$. The request $r_{f,\mathrm{OPT}}^{(i+1)}$ is released after schedule $S^{(i)}$ is started, i.e., after time $\alpha \cdot \mathrm{OPT}(t^{(i)})$. Thus,

$$
\mathrm{OPT}(t^{(i+1)}) \geq \alpha \cdot \mathrm{OPT}(t^{(i)}) + d(a_{f,\mathrm{OPT}}^{(i+1)}, b_{l,\mathrm{OPT}}^{(i)})
$$

$$
\overset{\triangle\text{-ineq}}{\geq} \alpha \cdot \mathrm{OPT}(t^{(i)}) + d(a_{f,\mathrm{OPT}}^{(i+1)}, p^{(i+1)}) - d(b_{l,\mathrm{OPT}}^{(i)}, O) - d(O, p^{(i+1)}). \quad (10)
$$

Since $S^{(i)}$ starts in O, ends in $p^{(i+1)}$ and serves $r_{l,\mathrm{OPT}}^{(i)}$, we obtain

$$
d(O, b_{l,\mathrm{OPT}}^{(i)}) + d(b_{l,\mathrm{OPT}}^{(i)}, O) \leq |S^{(i)}| + d(p^{(i+1)}, O) \overset{\text{Lem 1}}{\leq} \mathrm{OPT}(t^{(i)}) + d(p^{(i+1)}, O).
$$
$$(11)$$

Furthermore, because $\mathrm{OPT}[t^{(i+1)}]$ serves $r_{l,\mathrm{LAZY}}^{(i)}$ at $p^{(i+1)}$ before picking up $r_{f,\mathrm{OPT}}^{(i+1)}$ at $a_{f,\mathrm{OPT}}^{(i+1)}$, we have

$$
\mathrm{OPT}(t^{(i+1)}) \geq d(O, p^{(i+1)}) + d(p^{(i+1)}, a_{f,\mathrm{OPT}}^{(i+1)})
$$

$$
\overset{(7)}{>} d(O, p^{(i+1)}) + \alpha \cdot \mathrm{OPT}(t^{(i+1)}) - \mathrm{OPT}(t^{(i)}). \quad (12)
$$

Combining all of the above yields

$$
\mathrm{OPT}(t^{(i+1)}) \overset{(10),(11)}{\geq} \alpha \cdot \mathrm{OPT}(t^{(i)}) + d(a_{f,\mathrm{OPT}}^{(i+1)}, p^{(i+1)})
$$

$$
- \frac{\mathrm{OPT}(t^{(i)}) + d(p^{(i+1)}, O)}{2} - d(O, p^{(i+1)})
$$

$$
\overset{(12)}{>} \alpha \cdot \mathrm{OPT}(t^{(i)}) + d(a_{f,\mathrm{OPT}}^{(i+1)}, p^{(i+1)}) - \frac{\mathrm{OPT}(t^{(i)})}{2}
$$

$$
- \frac{3}{2}\left(\mathrm{OPT}(t^{(i)}) - (\alpha - 1)\mathrm{OPT}(t^{(i+1)})\right)
$$

$$
= \left(\frac{3}{2}\alpha - \frac{3}{2}\right)\mathrm{OPT}(t^{(i+1)}) - (2 - \alpha)\mathrm{OPT}(t^{(i)}) + d(a_{f,\mathrm{OPT}}^{(i+1)}, p^{(i+1)})
$$

$$
\overset{(7)}{>} \left(\frac{3}{2}\alpha - \frac{3}{2}\right)\mathrm{OPT}(t^{(i+1)}) - (2 - \alpha)\mathrm{OPT}(t^{(i)})
$$

$$
+ \alpha \cdot \mathrm{OPT}(t^{(i+1)}) - \mathrm{OPT}(t^{(i)})
$$

$$
= \left(\frac{5}{2}\alpha - \frac{3}{2}\right)\mathrm{OPT}(t^{(i+1)}) - (3 - \alpha)\mathrm{OPT}(t^{(i)})
$$

$$
\overset{\text{Obs 1a)}}{\geq} \left(\frac{5}{2}\alpha - \frac{3}{2}\right)\mathrm{OPT}(t^{(i+1)}) - \left(\frac{3}{\alpha} - 1\right)\mathrm{OPT}(t^{(i+1)})
$$

$$
= \left(\frac{5}{2}\alpha - \frac{1}{2} - \frac{3}{\alpha}\right)\mathrm{OPT}(t^{(i+1)})
$$

$$
\geq \mathrm{OPT}(t^{(i+1)})
$$

where the last inequality holds if and only if $\alpha \geq \frac{1}{10} \cdot (3 + \sqrt{129}) \approx 1.436$. As this is a contradiction, we have that $\mathrm{OPT}[t^{(i+1)}]$ serves all requests in $R^{(i)}$ before picking up $r_{f,\mathrm{OPT}}^{(i+1)}$ in $a_{f,\mathrm{OPT}}^{(i+1)}$. This completes the proof of the claim.

Now that we have established the claim, we turn back to the proof of Lemma 5. Let $T \geq 0$ denote the time it takes $\text{OPT}[t^{(i+1)}]$ until it has served $r_{l,\text{OPT}}^{(i)}$, i.e., all requests from $R^{(i)}$. First, observe that

$$T \geq \text{OPT}(t^{(i)}). \tag{13}$$

By the claim, we have

$$\text{OPT}(t^{(i+1)}) \geq T + d(b_{l,\text{OPT}}^{(i)}, a_{f,\text{OPT}}^{(i+1)}) + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})|. \tag{14}$$

The algorithm $\text{LAZY}(\alpha)$ finishes $R^{(i+1)}$ by time

$$
\begin{aligned}
t^{(i+1)} + S^{(i+1)} \overset{\text{Lem 4}}{=} & \; \alpha \cdot \text{OPT}(t^{(i)}) + |S^{(i)}| + |S^{(i+1)}| \\
\leq & \; \alpha \cdot \text{OPT}(t^{(i)}) + |S^{(i)}| + d(p^{(i+1)}, a_{f,\text{OPT}}^{(i+1)}) + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
\leq & \; \alpha \cdot \text{OPT}(t^{(i)}) + |S^{(i)}| + d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}) + d(b_{l,\text{OPT}}^{(i)}, a_{f,\text{OPT}}^{(i+1)}) \\
& \; + |S(R^{(i+1)}, a_{f,\text{OPT}}^{(i+1)})| \\
\overset{(14)}{\leq} & \; \alpha \cdot \text{OPT}(t^{(i)}) + |S^{(i)}| + d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}) + \text{OPT}(t^{(i+1)}) - T.
\end{aligned}
\tag{15}
$$

As $S^{(i)}$ visits $b_{l,\text{OPT}}^{(i)}$ before $p^{(i+1)}$ and $\text{OPT}[t^{(i+1)}]$ visits $p^{(i+1)}$ before $b_{l,\text{OPT}}^{(i)}$,

$$
\begin{aligned}
|S^{(i)}| + T \geq & \; \left(d(O, b_{l,\text{OPT}}^{(i)}) + d(b_{l,\text{OPT}}^{(i)}, p^{(i+1)}) \right) \\
& \; + \left(d(O, p^{(i+1)}) + d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}) \right) \\
= & \; 2 \cdot d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}) + d(O, b_{l,\text{OPT}}^{(i)}) + d(O, p^{(i+1)}) \\
\geq & \; 3 \cdot d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}).
\end{aligned}
\tag{16}
$$

Combined, we obtain that the algorithm finishes not later than

$$
\begin{aligned}
t^{(i+1)} + |S^{(i+1)}| \overset{(15)}{\leq} & \; \alpha \cdot \text{OPT}(t^{(i)}) + |S^{(i)}| + d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}) + \text{OPT}(t^{(i+1)}) - T \\
\overset{(16)}{\leq} & \; \alpha \cdot \text{OPT}(t^{(i)}) + |S^{(i)}| + \frac{|S^{(i)}| + T}{3} + \text{OPT}(t^{(i+1)}) - T \\
\overset{\text{Obs 1a)}}{\leq} & \; 2 \cdot \text{OPT}(t^{(i+1)}) + \frac{4}{3}|S^{(i)}| - \frac{2}{3}T \\
\overset{\text{Lem 1, (13)}}{\leq} & \; 2 \cdot \text{OPT}(t^{(i+1)}) + \frac{2}{3}\text{OPT}(t^{(i)}) \\
\overset{\text{Obs 1a)}}{\leq} & \; \left(2 + \frac{2}{3\alpha} \right) \cdot \text{OPT}(t^{(i+1)}) \\
\leq & \; (1 + \alpha) \cdot \text{OPT}(t^{(i+1)})
\end{aligned}
$$

where the last inequality holds if and only if $\alpha \geq \frac{1}{2} + \sqrt{11/12}$. $\square$

The above results enable us to prove Theorem 1.

Proof (of Theorem 1). Our goal was to prove by induction that every schedule is α-good for $\alpha \geq \frac{1}{2} + \sqrt{11/12}$. In Observation 2, we have proven the base case. In the induction step, we have distinguished several cases. First, we have seen in Observation 3 that the induction step holds if the previous schedule was interrupted. Next, we have seen in Lemma 1 that the induction hypothesis implies $|S^{(i+1)}| \leq \text{OPT}(t^{(i+1)})$. If the previous schedule was not interrupted, we have first seen in Lemma 2 that the induction step holds if $\text{OPT}[t^{(i+1)}]$ loads $r_{f,\text{OPT}}^{(i+1)}$ before serving $r_{l,\text{OPT}}^{(i)}$. If $\text{OPT}[t^{(i+1)}]$ serves $r_{l,\text{OPT}}^{(i)}$ before loading $r_{f,\text{OPT}}^{(i+1)}$, the induction step holds by Lemma 3 and Lemma 5. $\qquad\square$

5   Analysis on the half-line

In this section, we prove that LAZY is even better if the metric space considered is the half-line. In particular, we prove Theorem 2. To do this, we begin by showing that $\alpha + 1$ is an upper bound on the competitive ratio of LAZY(α) for $\alpha = \frac{1+\sqrt{3}}{2} \approx 1.366$. Later, we complement this upper bound with a lower bound construction and show that, for all $\alpha \geq 0$, LAZY(α) has a competitive ratio of at least $\frac{3+\sqrt{3}}{2} \approx 2.366$.

Since, for all $\alpha \geq \frac{1+\sqrt{3}}{2} \approx 1.366$, Observations 2 and 3, as well as Lemmas 1-4 hold, it remains to show a counterpart to Lemma 5 for $\alpha \geq \frac{1+\sqrt{3}}{2}$ on the half-line. Similarly to the proof of Theorem 1, combining Observations 2 and 3 and Lemmas 1-4 with Lemma 6 then yields Theorem 2.

Lemma 6. *Let $\frac{1+\sqrt{3}}{2} \leq \alpha \leq 2$, and let $M = \mathbb{R}_{\geq 0}$. Assume that the i-th schedule is α-good and was not interrupted, and that $\text{OPT}[t^{(i+1)}]$ serves $r_{l,\text{LAZY}}^{(i)}$ before collecting $r_{f,\text{OPT}}^{(i+1)}$. If (7) holds, then $t^{(i+1)} + |S^{(i+1)}| \leq (1+\alpha) \cdot \text{OPT}(t^{(i+1)})$.*

Proof. First, observe that, in Lemma 5, we have the same assumptions except that we worked on general metric spaces. Therefore, all the inequalities shown in Lemma 5 hold in this setting, too, so that we can use them for our proof. Next, note that on the half-line, we have for any $x, y \in M$

$$d(x,y) \leq \max\{d(x,O), d(y,O)\}. \tag{17}$$

We show that this implies that a similar claim as in Lemma 5 holds.
Claim: $\text{OPT}[t^{(i+1)}]$ *serves all requests in $R^{(i)}$ before picking up $r_{f,\text{OPT}}^{(i+1)}$ in $a_{f,\text{OPT}}^{(i+1)}$.*
To prove the claim, assume otherwise, i.e., that $\text{OPT}[t^{(i+1)}]$ serves $r_{l,\text{OPT}}^{(i)}$ after collecting $r_{f,\text{OPT}}^{(i+1)}$. The request $r_{f,\text{OPT}}^{(i+1)}$ is released after schedule $S^{(i)}$ is started, i.e., after time $\alpha \cdot \text{OPT}(t^{(i)})$. Thus,

$$\begin{aligned}
\text{OPT}(t^{(i+1)}) &\geq \alpha \cdot \text{OPT}(t^{(i)}) + d(a_{f,\text{OPT}}^{(i+1)}, b_{l,\text{OPT}}^{(i)}) \\
&\geq \alpha \cdot \text{OPT}(t^{(i)}) + d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}) - d(b_{l,\text{OPT}}^{(i)}, p^{(i+1)}) \\
&\overset{(17)}{\geq} \alpha \cdot \text{OPT}(t^{(i)}) + d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}) \\
&\quad - \max\{d(b_{l,\text{OPT}}^{(i)}, O), d(O, p^{(i+1)})\}.
\end{aligned} \tag{18}$$

Combining the above with the results from the proof of Lemma 5 yields

$$
\begin{aligned}
\text{OPT}(t^{(i+1)}) &\overset{(18),(11)}{\geq} \alpha \cdot \text{OPT}(t^{(i)}) + d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}) \\
&\qquad - \max\left\{ \frac{\text{OPT}(t^{(i)}) + d(p^{(i+1)}, O)}{2}, d(O, p^{(i+1)}) \right\} \\
&\overset{(12)}{>} \alpha \cdot \text{OPT}(t^{(i)}) + d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}) \\
&\qquad - \max\left\{ \frac{2\text{OPT}(t^{(i)}) - (\alpha-1)\text{OPT}(t^{(i+1)})}{2}, \right. \\
&\qquad\qquad\qquad\qquad \left. \text{OPT}(t^{(i)}) - (\alpha-1)\text{OPT}(t^{(i+1)}) \right\} \\
&= \frac{\alpha-1}{2}\text{OPT}(t^{(i+1)}) + (\alpha-1)\text{OPT}(t^{(i)}) + d(a_{f,\text{OPT}}^{(i+1)}, p^{(i+1)}) \\
&\overset{(7)}{>} \left(\frac{\alpha-1}{2} + \alpha \right)\text{OPT}(t^{(i+1)}) + (\alpha-2)\text{OPT}(t^{(i)}) \\
&\overset{\text{Obs 1a)}}{\geq} \left(\frac{\alpha-1}{2} + \alpha + \frac{\alpha-2}{\alpha} \right)\text{OPT}(t^{(i+1)}) \\
&\geq \text{OPT}(t^{(i+1)})
\end{aligned}
$$

where the last inequality holds for all $\alpha \geq \frac{4}{3}$. As this is a contradiction, we have that $\text{OPT}[t^{(i+1)}]$ serves all requests in $R^{(i)}$ before picking up $r_{f,\text{OPT}}^{(i+1)}$ in $a_{f,\text{OPT}}^{(i+1)}$. This completes the proof of the claim.

Now that we have established the claim, we turn back to the proof of Lemma 6. Let $T \geq 0$ denote the time it takes $\text{OPT}[t^{(i+1)}]$ until it has served $r_{l,\text{OPT}}^{(i)}$, i.e., all requests from $R^{(i)}$. First, observe that

$$T \geq \text{OPT}(t^{(i)}). \tag{19}$$

If $p^{(i+1)} \geq b_{l,\text{OPT}}^{(i)}$, as $\text{OPT}[t^{(i+1)}]$ visits $p^{(i+1)}$ before $b_{l,\text{OPT}}^{(i)}$, we have

$$T \geq d(O, p^{(i+1)}) + d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}) \overset{(17)}{\geq} 2 \cdot d(p^{(i+1)}, b_{l,\text{OPT}}^{(i)}).$$

Otherwise, if $p^{(i+1)} < b_{l,\text{OPT}}^{(i)}$, as $S^{(i)}$ visits $b_{l,\text{OPT}}^{(i)}$ before $p^{(i+1)}$, we have

$$T \overset{(19)}{\geq} \text{OPT}(t^{(i)}) \geq |S^{(i)}| \geq d(O, b_{l,\text{OPT}}^{(i)}) + d(b_{l,\text{OPT}}^{(i)}, p^{(i+1)}) \overset{(17)}{\geq} 2 \cdot d(b_{l,\text{OPT}}^{(i)}, p^{(i+1)}).$$

Thus, in either case, we have

$$d(b_{l,\text{OPT}}^{(i)}, p^{(i+1)}) \leq \frac{T}{2}. \tag{20}$$

Combined, we obtain that the algorithm finishes not later than

$$
\begin{aligned}
t^{(i+1)} + |S^{(i+1)}| &\overset{(15)}{\leq} \alpha \cdot \mathrm{OPT}(t^{(i)}) + |S^{(i)}| + d(p^{(i+1)}, b^{(i)}_{l,\mathrm{OPT}}) + \mathrm{OPT}(t^{(i+1)}) - T \\
&\overset{(20)}{\leq} \alpha \cdot \mathrm{OPT}(t^{(i)}) + |S^{(i)}| + \frac{T}{2} + \mathrm{OPT}(t^{(i+1)}) - T \\
&\overset{\text{Obs 1a)}}{\leq} 2 \cdot \mathrm{OPT}(t^{(i+1)}) + |S^{(i)}| - \frac{T}{2} \\
&\overset{\text{Lem 1, (19)}}{\leq} 2 \cdot \mathrm{OPT}(t^{(i+1)}) + \frac{1}{2}\mathrm{OPT}(t^{(i)}) \\
&\overset{\text{Obs 1a)}}{\leq} \left(2 + \frac{1}{2\alpha}\right) \cdot \mathrm{OPT}(t^{(i+1)}) \\
&\leq (1 + \alpha) \cdot \mathrm{OPT}(t^{(i+1)})
\end{aligned}
$$

where the last inequality holds if and only if $\alpha \geq \frac{1+\sqrt{3}}{2} \approx 1.366$.   □

5.1   Lower bound on the half-line

In this section, we prove Theorem 3, i.e., we show that $\mathrm{LAZY}(\alpha)$ has a competitive ratio of at least $\frac{3+\sqrt{3}}{2} \approx 2.366$ for every choice of the parameter α, even when the metric space is the half line. Our proof builds on some lower bound constructions given in [5]. We begin by restating needed results.

Lemma 7 ([5]). $\mathrm{LAZY}(\alpha)$ *has competitive ratio at least* $1+\alpha$ *for the open online dial-a-ride problem on the half-line for all* $\alpha \geq 0$ *and every capacity* $c \in \mathbb{N} \cup \{\infty\}$.

The following bound holds, since [5, Proposition 2] only uses the half line.

Lemma 8 ([5]). *For every* $\alpha \in [0,1)$, $\mathrm{LAZY}(\alpha)$ *has a competitive ratio of at least* $1 + \frac{3}{\alpha+1} > 2.5$ *for the open online dial-a-ride problem on the half-line for every capacity* $c \in \mathbb{N} \cup \{\infty\}$.

Combining these two results gives that, for $\alpha \in [0,1) \cup [\frac{1+\sqrt{3}}{2}, \infty)$, the competitive ratio of $\mathrm{LAZY}(\alpha)$ is at least $\frac{3+\sqrt{3}}{2} \approx 2.366$.

The next proposition closes the gap between $\alpha < 1$ and $\alpha \geq 1.366$ and completes the proof of Theorem 3. An overview of the lower bounds for different domains of α can be found in Figure 2.

Proposition 1. *For every* $\alpha \in [1, 1.366)$, *the algorithm* $\mathrm{LAZY}(\alpha)$ *has a competitive ratio of at least* $2 + \frac{1}{2\alpha}$ *for the open online dial-a-ride problem on the half-line for every capacity* $c \in \mathbb{N} \cup \{\infty\}$.

Proof. Let $\alpha \in [1, 1.366)$ and let $\varepsilon > 0$ be sufficiently small. We define an instance (cf. Figure 1) by giving the request sequence

$$
\begin{aligned}
&r_1 = (0, 1; 0)\,, \; r_2 = (1, 0; 0)\,, \; r_3 = (1, 2 - \varepsilon; 0), \\
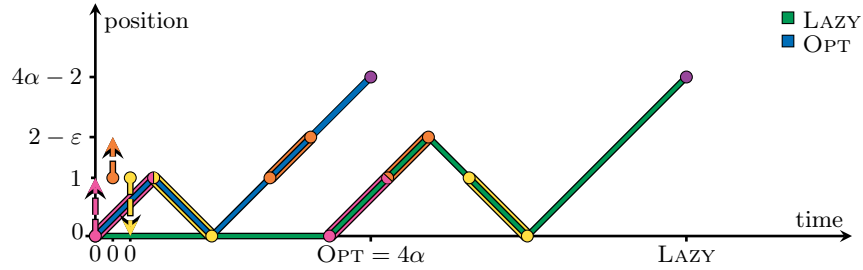&\text{and } r_4 = (4\alpha - 2, 4\alpha - 2; 4\alpha).
\end{aligned}
$$

Fig. 1. Instance of the open online dial-a-ride problem on the half-line where $\textsc{Lazy}(\alpha)$ has a competitive ratio of at least $2 + \frac{1}{2\alpha}$ for all $\alpha \in [1, 1.366)$.

The offline optimum delivers the requests in the order (r_1, r_2, r_3, r_4) with no waiting times. This takes 4α time units.

On the other hand because $\textsc{Opt}(0) = 4 - 2\varepsilon$, $\textsc{Lazy}(\alpha)$ waits in the origin until time $\alpha(4 - 2\varepsilon)$ and starts serving requests r_1, r_2, r_3 in the order (r_1, r_3, r_2). At time 4α, request r_4 is released. At this time, serving the loaded request r_1 and returning to the origin takes time

$$\alpha(4 - 2\varepsilon) + 2 = 4\alpha + 2 - 2\alpha\varepsilon \overset{\alpha \in [1,1.366), \varepsilon \ll 1}{>} 4\alpha^2 = \alpha \cdot \textsc{Opt}(4\alpha).$$

Thus, $\textsc{Lazy}(\alpha)$ continues its schedule and afterwards serves r_4. Overall, this takes time (at least) $\alpha(4 - 2\varepsilon) + (4 - 2\varepsilon) + 4\alpha - 2 = 8\alpha + 2 - (2\alpha + 2)\varepsilon$. Letting $\varepsilon \to 0$, we obtain that the competitive ratio of $\textsc{Lazy}(\alpha)$ is at least

$$\lim_{\varepsilon \to 0} \frac{8\alpha + 2 - (2\alpha + 2)\varepsilon}{4\alpha} = 2 + \frac{1}{2\alpha}.$$

$\square$

A   Factor-revealing approach for the half-line

We show how to use the factor revealing approach from Section 3 for the dial-a-ride problem on the half-line. Consider the following variables (recall that $k \in \mathbb{N}$ is the number of schedules started by $\textsc{Lazy}(\alpha)$).

- $t_1 = t^{(k-1)}$, the start time of the second to last schedule
- $t_2 = r^{(k)}$, the start time of the last schedule
- $s_1 = |S^{(k-1)}|$, the duration of the second to last schedule
- $s_2 = |S^{(k)}|$, the duration of the last schedule
- $\textsc{Opt}_1 = \textsc{Opt}(t^{(k-1)})$, duration of the optimal tour serving requests released until $t^{(k-1)}$
- $\textsc{Opt}_2 = \textsc{Opt}(t^{(k)})$, duration of the optimal tour
- $p_1 = p^{(k)}$, the position where $\textsc{Lazy}(\alpha)$ ends the second to last schedule
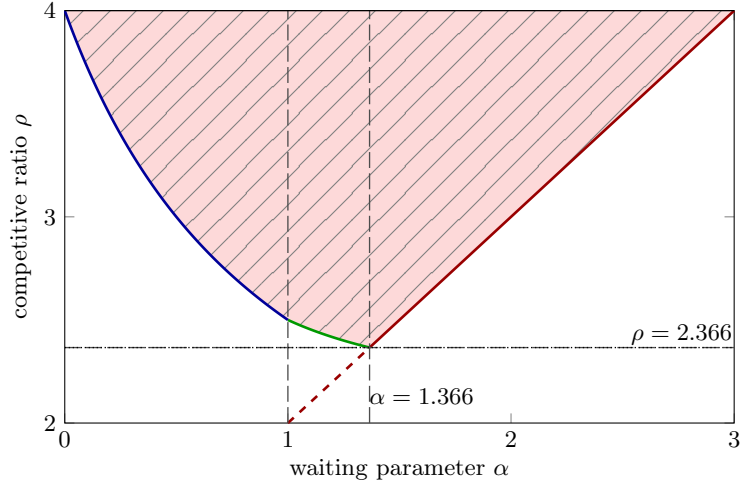
Fig. 2. Lower bounds on the competitive ratio of $\textsc{Lazy}(\alpha)$ depending on α. The lower bound of Lemma 7 is depicted in red, the lower bound of Lemma 8 in blue, and the lower bound of Proposition 1 in green.

- $p_2 = a_{f,\textsc{Opt}}^{(k)}$, the position of the first request in $R^{(k)}$ picked up first by the optimal tour

- $s_2^a = |S(R^{(k)}, a_{f,\textsc{Opt}}^{(k)})|$, duration of the schedule serving $R^{(k)}$ starting in p_2

- $d = d(p^{(k)}, a_{f,\textsc{Opt}}^{(k)})$, the distance between p_1 and p_2

With these variables

$$x = \left(t_1, t_2, s_1, s_2, \textsc{Opt}_1, \textsc{Opt}_2, p_1, p_2, s_2^a, d\right),$$

we can create the following valid optimization problem.

$$
\begin{aligned}
\max \quad & t_2 + s_2 \\
\text{s.t.} \quad & \text{OPT}_2 = 1 & & (21) \\
& d = |p_1 - p_2| & & (22) \\
& t_2 = \max\{t_1 + s_1, \alpha \text{OPT}_2\} & & (23) \\
& t_1 \geq \alpha \text{OPT}_1 & & (24) \\
& \text{OPT}_1 \geq p_1 & & (25) \\
& s_2 \leq d + s_2^a & & (26) \\
& \text{OPT}_2 \geq t_1 + s_2^a & & (27) \\
& t_1 + s_1 \leq (1 + \alpha)\text{OPT}_1 & & (28) \\
& \text{OPT}_2 \geq p_1 + d \quad \text{or} \quad \text{OPT}_2 \geq t_1 + d & & (29) \\
& d \geq \alpha \text{OPT}_2 - \text{OPT}_1 \quad \text{or} \quad s_1 - p_1 \leq 2(\text{OPT}_2 - p_2) & & (30) \\
& x \geq 0 & & (31)
\end{aligned}
$$

Note that in (29) and (30), at least one of the two inequalities has to be satisfied in each case. In order to obtain an MILP, one can introduce four binary variables $b_1, \ldots, b_4$ to model constraints (22), (23), (29), and (30).

With $M > 0$ being a large enough constant, equality (22) can be replaced by the inequalities

$$
\begin{aligned}
d &\geq p_1 - p_2, \\
d &\geq p_2 - p_1, \\
d &\leq p_1 - p_2 + b_1 \cdot M, \\
d &\leq p_2 - p_1 + (1 - b_1) \cdot M.
\end{aligned}
$$

Equality (23) can be replaced by the inequalities

$$
\begin{aligned}
t_2 &\geq t_1 + s_1, \\
t_2 &\geq \alpha \text{OPT}_2, \\
t_2 &\leq t_1 + s_1 + b_2 \cdot M, \\
t_2 &\leq \alpha \text{OPT}_2 + (1 - b_2) \cdot M.
\end{aligned}
$$

Constraint (29) can be replaced by the inequalities

$$
\begin{aligned}
\text{OPT}_2 &\geq p_1 + d - b_3 \cdot M, \\
\text{OPT}_2 &\geq t_1 + d - (1 - b_3) \cdot M,
\end{aligned}
$$

and, likewise, (30) by the inequalities

$$
\begin{aligned}
d &\geq \alpha \text{OPT}_2 - \text{OPT}_1 - b_4 \cdot M, \\
s_1 - p_1 &\leq 2(\text{OPT}_2 - p_2) + (1 - b_4) \cdot M.
\end{aligned}
$$

The resulting MILP has the optimal solution

$$\big(t_1, t_2, s_1, s_2, \mathrm{OPT}_1, \mathrm{OPT}_2, p_1, p_2, s_2^a, d, b_1, b_2, b_3, b_4\big)$$
$$= \Big(1, \frac{\alpha+1}{\alpha}, \frac{1}{\alpha}, 2-\alpha, \frac{1}{\alpha}, 1, 0, 2-\alpha, 0, 2-\alpha, 0, 1, 1, 1\Big)$$

and optimal value $\max\{3 + \frac{1}{\alpha} - \alpha, 1 + \alpha\}$. For $\alpha = \frac{1+\sqrt{3}}{2} > 1.366$, this expression is minimized.

References

1. Ascheuer, N., Krumke, S.O., Rambau, J.: Online dial-a-ride problems: Minimizing the completion time. In: Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS). pp. 639–650 (2000)
2. Ausiello, G., Demange, M., Laura, L., Paschos, V.: Algorithms for the on-line quota traveling salesman problem. Information Processing Letters **92**(2), 89–94 (2004)
3. Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L., Talamo, M.: Algorithms for the on-line travelling salesman. Algorithmica **29**(4), 560–581 (2001)
4. Ausiello, G., Allulli, L., Bonifaci, V., Laura, L.: On-line algorithms, real time, the virtue of laziness, and the power of clairvoyance. In: Proceddings of the 3rd International Conference on Theory and Applications of Models of Computation (TAMC). pp. 1–20 (2006)
5. Baligács, J., Disser, Y., Mosis, N., Weckbecker, D.: An improved algorithm for open online dial-a-ride. In: Proceedings of the 20th Workshop on Approximation and Online Algorithms (WAOA) (2022)
6. Bienkowski, M., Kraska, A., Liu, H.: Traveling repairperson, unrelated machines, and other stories about average completion times. In: Proceedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP). pp. 28:1–28:20 (2021)
7. Bienkowski, M., Liu, H.: An improved online algorithm for the traveling repair-person problem on a line. In: Proceedings of the 44th International Symposium on Mathematical Foundations of Computer Science (MFCS). pp. 6:1–6:12 (2019)
8. Birx, A.: Competitive analysis of the online dial-a-ride problem. Ph.D. thesis, TU Darmstadt (2020)
9. Birx, A., Disser, Y.: Tight analysis of the smartstart algorithm for online dial-a-ride on the line. SIAM Journal on Discrete Mathematics **34**(2), 1409–1443 (2020)
10. Birx, A., Disser, Y., Schewior, K.: Improved bounds for open online dial-a-ride on the line. Algorithmica (2022)
11. Bjelde, A., Disser, Y., Hackfeld, J., Hansknecht, C., Lipmann, M., Meißner, J., Schewior, K., Schlöter, M., Stougie, L.: Tight bounds for online tsp on the line. ACM Transactions on Algorithms **17**(1) (2020)
12. Blom, M., Krumke, S.O., de Paepe, W.E., Stougie, L.: The online TSP against fair adversaries. INFORMS Journal on Computing **13**(2), 138–148 (2001)
13. Bonifaci, V., Stougie, L.: Online k-server routing problems. Theory of Computing Systems **45**(3), 470–485 (2008)
14. Feuerstein, E., Stougie, L.: On-line single-server dial-a-ride problems. Theoretical Computer Science **268**(1), 91–105 (2001)
15. Hauptmeier, D., Krumke, S., Rambau, J., Wirth, H.C.: Euler is standing in line dial-a-ride problems with precedence-constraints. Discrete Applied Mathematics **113**(1), 87–107 (2001)

16. Hauptmeier, D., Krumke, S.O., Rambau, J.: The online dial-a-ride problem under reasonable load. In: Proceedings of the 4th Italian Conference on Algorithms and Complexity (CIAC). pp. 125–136 (2000)
17. Jaillet, P., Lu, X.: Online traveling salesman problems with service flexibility. Networks **58**(2), 137–146 (2011)
18. Jaillet, P., Lu, X.: Online traveling salesman problems with rejection options. Networks **64**(2), 84–95 (2014)
19. Jaillet, P., Wagner, M.R.: Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. Operations Research **56**(3), 745–757 (2008)
20. Jawgal, V.A., Muralidhara, V.N., Srinivasan, P.S.: Online travelling salesman problem on a circle. In: In Proceedings of the 15th International Conference on Theory and Applications of Models of Computation (TAMC). pp. 325–336 (2019)
21. Krumke, S.O.: Online optimization competitive analysis and beyond. Habilitation thesis, Zuse Institute Berlin (2001)
22. Krumke, S.O., Laura, L., Lipmann, M., Marchetti-Spaccamela, A., de Paepe, W., Poensgen, D., Stougie, L.: Non-abusiveness helps: An O(1)-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem. In: Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX). pp. 200–214 (2002)
23. Krumke, S.O., de Paepe, W.E., Poensgen, D., Lipmann, M., Marchetti-Spaccamela, A., Stougie, L.: On minimizing the maximum flow time in the online dial-a-ride problem. In: Proceedings of the 3rd International Conference on Approximation and Online Algorithms (WAOA). pp. 258–269 (2006)
24. Krumke, S.O., de Paepe, W.E., Poensgen, D., Stougie, L.: News from the online traveling repairman. Theoretical Computer Science **295**(1-3), 279–294 (2003)
25. Lipmann, M.: On-line routing. Ph.D. thesis, Technische Universiteit Eindhoven (2003)
26. Lipmann, M., Lu, X., de Paepe, W.E., Sitters, R.A., Stougie, L.: On-line dial-a-ride problems under a restricted information model. Algorithmica **40**(4), 319–329 (2004)