Improved Bounds for Online TSP on the Half-Line

Júlia Baligács
University of Warsaw, Poland

Yann Disser
TU Darmstadt, Germany

Linda Thelen
TU Darmstadt, Germany

Abstract

In the open online traveling salesperson problem, requests appear over time at different positions of a metric space. A single agent traveling at unit speed must serve all requests, by visiting their positions, with the objective of minimizing the completion time. We propose online algorithms for this problem for the case that the metric space is the half-line. First, we present a 2-competitive algorithm in which the server always either moves at unit speed or remains stationary, which improves on the previously best-known ratio of 2.04. We further observe that algorithms must sometimes move slower than unit speed to achieve competitive ratios below 2. We introduce a second algorithm that beats the barrier of 2 by carefully modulating its speed, and prove a tight bound of 1.75 on its competitive ratio. Finally, we slightly strengthen a known lower bound on the best-possible competitive ratio on the half-line from 1.627 to 1.646.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, competitive analysis, server problems, online TSP

Digital Object Identifier 10.4230/LIPIcs.SWAT.2026.4

Funding *Júlia Baligács*: Supported by the project BOBR that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 948057).

1 Introduction

We study the online traveling salesperson problem (TSP) on the half-line. In this problem, requests of the form $r = (t, p)$ are revealed over time, where $t \in \mathbb{R}_{\geq 0}$ denotes the release time and $p \in \mathbb{R}_{\geq 0}$ the position of the request. A single server, initially located at the origin and moving at speed at most 1, must serve every request by visiting position p at some time not earlier than time t. In the online setting, the algorithm only learns about a request at its release time, whereas an offline algorithm knows the entire request sequence from time 0. The objective is to minimize the makespan, i.e., the time when all requests are served. Note that the offline setting does not coincide with the classical TSP, as the offline algorithm is still restricted to serving a request after its release time. Furthermore, we consider the *open* version of the problem, where, in contrast to the *closed* version, the server *does not* need to return to the origin.

Online TSP models dynamic routing scenarios arising, for instance, in robotic maintenance, cleaning, or collection tasks where jobs appear over time. In many such settings, the depot lies at the boundary of a one-dimensional operating region, such as inspection along a track, pipeline, or corridor, so that the underlying metric space is the half-line, making it a natural case to study. While the online TSP has been extensively studied in many variants [3, 4, 5, 12, 18, 19, 20, 25], in particular for the underlying metric space being the real line $\mathbb{R}$ or higher-dimensional Euclidean spaces $\mathbb{R}^n$, we focus here on algorithms tailored specifically to the half-line $\mathbb{R}_{\geq 0}$. Although this setting appears simple at first glance,

obtaining optimal strategies turns out to be surprisingly challenging. The half-line has fundamentally different structural properties due to the lack of symmetry around the origin, and it seems that optimal strategies cannot be obtained by simple adaptations of known algorithms. For instance, we will see that, unlike any algorithm previously proposed for online TSP, an optimal online strategy on the half-line must already start moving before the first request is revealed.

As usual in online optimization, we measure the quality of a (deterministic) algorithm in terms of *competitive analysis*. For an algorithm ALG and a request sequence σ, let $\text{ALG}(\sigma)$ denote the completion time of ALG on σ, and let $\text{OPT}(\sigma)$ denote the completion time of an optimal offline solution. We say that ALG is *ρ-competitive* if $\text{ALG}(\sigma) \leq \rho\,\text{OPT}(\sigma)$ for every request sequence σ. The *competitive ratio of* ALG is the infimum over all such ρ, and the *competitive ratio of the problem* is the best ratio achievable by any online algorithm.

Throughout the paper, we assume without loss of generality that every request $r = (t, p)$ satisfies $t \geq p$. Indeed, since the server moves at speed at most 1, even the offline optimum cannot reach position p before time p, and is therefore unaffected by this assumption. Moreover, an online algorithm could only benefit from learning about a request earlier. Hence, this assumption is justified when the request sequence is constructed adversarially and does not affect the competitive ratio of the problem.

Finally, we introduce the following notational conventions. Given an online algorithm ALG, we write $\text{pos}(t) \in \mathbb{R}_{\geq 0}$ for the position of the server at time t. For a request sequence σ, we sometimes abbreviate the completion times $\text{ALG}(\sigma)$ and $\text{OPT}(\sigma)$ by ALG and OPT when the sequence is clear from context. At any point in time, the server can move in (at most) two directions. We refer to the direction towards the origin as *left* and to the direction away from the origin as *right*.

Our results

We begin by introducing the algorithm LEFTFIRST (cf. Algorithm 1), which follows a very simple strategy: it prioritizes requests that lie to the server's left over those to its right. We show that this algorithm achieves a competitive ratio of 2, improving on the previously best known ratio of 2.04 [11]. We further observe that no algorithm restricted to moving either at full speed or not at all can achieve a better competitive ratio.

▶ **Theorem 1.** *The algorithm* LEFTFIRST *is* 2*-competitive for online TSP on the half-line. Moreover, this ratio is optimal among all algorithms in which the server is restricted to speeds* 0 *and* 1*.*

In particular, the theorem shows that fundamentally new ideas are required to beat the barrier of 2. Our main contribution is to propose such an idea in the form of a parameterized algorithm $\text{LEFTGUARD}(\alpha)$ for $\alpha \in (0, 1]$ (cf. Algorithm 2). Its strategy can be summarized as follows. The algorithm maintains at all times that $\text{pos}(t) \leq \alpha t$, enabling the server to reach requests to its left in a timely manner. It assigns strict deadlines to such requests and prioritizes requests to its right if and only if these deadlines can still be met. Our main result is a careful analysis of this algorithm. We show that, for $\alpha = 0.75$, it achieves a competitive ratio of 1.75. After identifying the most difficult instances for LEFTGUARD, a key ingredient of our proof is an LP-duality argument that yields sharp performance bounds. Moreover, we prove that this analysis is tight in the sense that $\text{LEFTGUARD}(\alpha)$ has competitive ratio at least 1.75 for every choice of α.

▶ **Theorem 2.** $\text{LEFTGUARD}(0.75)$ *is 1.75-competitive for open online TSP on the half-line. Moreover, the choice* $\alpha = 0.75$ *is optimal.*

We complement our result by a general lower bound on the best-possible competitive ratio that slightly improves the known lower bound of 1.627 [26].

▶ **Theorem 3.** *Every algorithm for open online TSP on the half-line has competitive ratio at least 1.6463.*

Related work

While this is, to the best of our knowledge, the first work to specifically focus on open online TSP on the half-line, the problem has been studied before on other metric spaces. Most notably, Bjelde et al. tightly analyzed the problem on the line with a competitive ratio of 2.04 [11]. Their upper bound also applies to the special case of the half-line and, prior to our work, was the best known upper bound in this setting. Their lower bound extends to general metric spaces and remains the best known lower bound there. The best known algorithm for general metric spaces, introduced by Bonifaci and Stougie, achieves a competitive ratio of $1 + \sqrt{2} \approx 2.41$ [13].

By contrast, the closed version of the problem is tightly analyzed in all three cases: It has a competitive ratio of 2 on general metric spaces [5], of 1.64 on the line [5, 11] and of 1.5 on the half-line [12]. The optimal algorithm for the closed version on the half-line follows a strategy that can be viewed as the "opposite" of our algorithm LeftFirst: That algorithm moves right whenever a request to the server's right is available and otherwise moves left.

Online TSP has also been investigated in several related settings, for instance when the metric space is a circle [20], when distances are asymmetric [3], or when the request sequence satisfies additional structural assumptions [12, 25].

The online TSP can be viewed as a special case of the extensively studied *online dial-a-ride problem* [2, 7, 9, 10, 17, 24, 26], where each requests is a transportation request consisting of a release time, a starting position, and a destination. In other words, the online TSP corresponds to the special case of online dial-a-ride in which starting position and destination always coincide. Classical online dial-a-ride strategies include Ignore, which repeatedly recomputes a schedule for the currently unserved requests [8, 23]; Replan, which greedily recomputes an optimal tour whenever a new request arrives [5, 8, 23]; and Lazy, which deliberately delays service to aggregate requests before executing a tour [7, 26].

Other online variants of the TSP focus on different types of uncertainty. In *online graph exploration* [6, 15, 16, 22], the metric space is initially unknown and all vertices must be visited. In *universal TSP* [14, 21], the goal is to find a fixed master tour that is competitive for any subset of active requests. Recently, Abrahamsen et al. [1] introduced a model where requests are revealed one-by-one and must be irrevocably sorted to minimize total travel distance, rather than minimizing the makespan for requests arriving over time.

2 Warm-up: moving at speed 1 or 0

Before considering the problem in full-generality, we first consider the easier case where the server is only allowed to move with full speed or not move at all. We first observe a simple lower bound on the competitive ratio of this variant of the problem.

▶ **Lemma 4.** *If the server is only allowed to move at speed 1 or 0, the competitive ratio of open online TSP on the half-line is at least 2. The same lower bound applies when the server is not allowed to move before the first request is revealed.*

Algorithm 1 LeftFirst.

```
if there is a left-request:
    MoveLeft
else
    if there is a right-request:
        MoveRight
    else
        Stay
```

Proof. Fix an algorithm Alg in which the server can move only at speed 0 or 1. Then there exists some time $\varepsilon > 0$ such that, before time ε, the server has not changed its speed. Hence $\text{pos}(\varepsilon) \in \{0, \varepsilon\}$. If $\text{pos}(\varepsilon) = 0$, we release the request $r = (\varepsilon, \varepsilon)$, and if $\text{pos}(\varepsilon) = \varepsilon$, we release the request $r' = (\varepsilon, 0)$. In both cases, we have $\text{Opt} = \varepsilon$, while Alg cannot complete the request before time 2ε. Therefore, the competitive ratio is at least 2. Observe that even if the server is allowed to move at intermediate speeds, as long as it does not move before the first request is revealed, the same argument still applies. ◀

Next, we give an algorithm whose competitive ratio matches this lower bound.

2.1 A 2-competitive algorithm

When considering algorithms for online TSP on the half-line (or even on the line), observe that, at any point in time, the algorithm only needs to consider at most two requests: the request with the leftmost position among all unserved requests to the algorithm's left (referred to as the *left-request*) and the request with the rightmost position among all unserved requests to the algorithm's right (referred to as the *right-request*). At some times, there may be no left-request or no right-request. This is the case when all unserved requests lie on the same side of the server. If there are multiple requests at the same position, we select the one with the latest release time. By serving the left- and right-request, the algorithm necessarily also serves all other requests on the way. In our analysis, we therefore consider only left- and right-requests. The online algorithm is thus defined by a strategy that determines when to serve the left-request, when to serve the right-request, and when to wait.

The algorithm LeftFirst is formally defined in Algorithm 1 and follows a simple strategy: It prioritizes left-requests over right-requests. More precisely, whenever a left-request is available, the server executes command MoveLeft, i.e., travels towards the origin at full speed (note that MoveLeft is never invoked when the server is located in the origin). If no left-request is available but a right-request exists, the server executes command MoveRight, i.e., it moves away from the origin at full speed. If there are no unserved requests, the server follows command Stay, i.e., remains stationary.

Before turning to the analysis of the algorithm, we state a simple observation that will be useful throughout our paper. Since the offline optimum cannot serve requests before they are released, the following holds.

▶ **Observation 5.** *If t is the release time of any request of the input sequence, then* $\text{Opt} \geq t$.

Now we have all the prerequisites at hand to analyze the algorithm LeftFirst.

▶ **Theorem 6.** LeftFirst *is 2-competitive.*

Proof. Fix an arbitrary request sequence. Let $r = (t, p)$ denote the last request served by LeftFirst. First, observe that, if r is a left-request, we have

$$\textsc{LeftFirst} \leq t + (\mathrm{pos}(t) - p) \leq t + \mathrm{pos}(t) \leq 2t \overset{\text{Obs. 5}}{\leq} 2\textsc{Opt},$$

where we have made used of the fact that $\mathrm{pos}(t) \leq t$ since the server cannot move at speed more than 1. Therefore, we can assume from now on that r is a right-request.

Next, observe that, if the server does not move left after time t, we have

$$\textsc{LeftFirst} \leq t + (p - \mathrm{pos}(t)) \leq t + p \leq 2t \overset{\text{Obs. 5}}{\leq} 2\textsc{Opt},$$

where we have used the assumption that every request fulfills $t \geq p$.

Therefore, we can assume from now on that there was an unserved left-request after time t and let $r_L = (t_L, p_L)$ denote the last left-request served by LeftFirst. Observe that we have

$$\textsc{LeftFirst} \leq t_L + (\mathrm{pos}(t_L) - p_L) + (p - p_L). \tag{1}$$

We distinguish now the following two cases: (1) The offline optimum serves r_L before r or (2) The offline optimum serves r before r_L. In the first case, we have

$$\textsc{Opt} \geq t_L + (p - p_L). \tag{2}$$

With (1) and (2), we obtain

$$\textsc{LeftFirst} \leq t_L + (\mathrm{pos}(t_L) - p_L) + (p - p_L) = \underbrace{t_L + p - p_L}_{\leq \textsc{Opt}} + \underbrace{\mathrm{pos}(t_L)}_{\leq t_L \leq \textsc{Opt}} - \underbrace{p_L}_{\geq 0} \leq 2\textsc{Opt}.$$

In the second case, i.e., where Opt serves r before r_L, we have

$$\textsc{Opt} \geq t + (p - p_L). \tag{3}$$

Moreover, observe that we have $\mathrm{pos}(t_L) \leq t$ because either $t \geq t_L \geq \mathrm{pos}(t_L)$ or $t_L \geq t$ and therefore $r = (t, p)$ was an unserved right-request at time t_L. With (1) and (3), we obtain

$$\textsc{LeftFirst} \leq t_L + \underbrace{\mathrm{pos}(t_L)}_{\leq t} - \underbrace{p_L}_{\geq 0} + p - p_L \leq \underbrace{t_L}_{\leq \textsc{Opt}} + \underbrace{t + p - p_L}_{\leq \textsc{Opt}} \leq 2\textsc{Opt}.$$

To summarize, we have in either case that LeftFirst ≤ 2Opt, which completes the proof of the theorem. ◀

Note that Theorem 6 and Lemma 4 together complete the proof of Theorem 1.

3 Improved algorithm for open online TSP on the half-line

In this section, we prove our main result (Theorem 2), i.e., we prove that the competitive ratio of open online TSP on the half-line is at most 1.75. For this, we begin with introducing the algorithm LeftGuard.

Algorithm 2 LeftGuard(α).

```
if there is a left-request r_L = (t_L, p_L):
    if there is a right-request r_R = (t_R, p_R) with
      max{t + p_R − pos(t), p_R/α} + p_R − p_L ≤ (1+α)t_L:
        MoveRight(α)
    else
        MoveLeft
else
    MoveRight(α)
```

Algorithm MoveRight(α).

```
if pos(t) < αt:
    move right at speed 1
else
    move right at speed α
```

Algorithm MoveLeft.

```
move left at speed 1
```

3.1 The algorithm LeftGuard

The algorithm LeftGuard is formally defined in Algorithm 2. It is parameterized by a value $\alpha \in (0, 1]$ and we let t denote the current time. Its intuition can be summarized as follows: First, recall that no algorithm is better than 2-competitive if the server does not move while being idle (Lemma 4), so we let the server move at speed α before the first request is revealed. As before, we consider always at most two requests – the left-request and the right-request. The most important property of our algorithm is that every left-request $r_L = (t_L, p_L)$ has a strict deadline, meaning that it must be served no later than at time $(1 + \alpha)t_L$. To make this possible, the server maintains at every time t that $\text{pos}(t) \leq \alpha t$. Therefore, whenever a new left-request $r_L = (t_L, p_L)$ appears, the server is located at $\text{pos}(t_L) \leq \alpha t_L$ so that, by moving left immediately, position $p_L \geq 0$ can be reached by time $(1 + \alpha)t_L$. However, if the server can serve a right-request without violating the deadlines of left-requests, it proceeds to do so. Note that a right-request $r_R = (t_R, p_R)$ can be reached by time $\max\{t + p_R - \text{pos}(t), \frac{p_R}{\alpha}\}$, where the first term corresponds to moving from $\text{pos}(t)$ to p_R at unit speed, and the second term is the arrival time at p_R in case $\text{pos}(t) \leq \alpha t$ becomes tight. Therefore, the server prioritizes a right-request over a left-request if and only if $\max\{t + p_R - \text{pos}(t), \frac{p_R}{\alpha}\} + p_R - p_L \leq (1 + \alpha)t_L$.

The algorithm definition contains the subroutines MoveLeft and MoveRight(α). When MoveLeft is invoked, it simply orders the server to move towards the origin at unit speed (observe that MoveLeft is never invoked when the server is located at the origin). MoveRight(α) depends on the parameter α and orders the server to move away from the origin at full speed maintaining that $\text{pos}(t) \leq \alpha t$. In other words, the server moves at speed 1 as long as $\text{pos}(t) < \alpha t$ and at speed α when $\text{pos}(t) = \alpha t$. In particular, the server always moves either at speed 1 or at speed α. When it moves at speed α, we say that it *slows down*.

Let us summarize the most important properties that immediately follow from the algorithm's definition.

▶ **Observation 7.** LeftGuard(α) *fulfills the following properties.*

i) *At every time t, the server position fulfills* $\text{pos}(t) \leq \alpha t$.

ii) *Every left-request $r_L = (t_L, p_L)$ is served no later than at time $(1 + \alpha)t_L$, unless a new left-request appeared before time $(1 + \alpha)t_L$.*

iii) *Every right request $r_R = (t_R, p_R)$ is served no earlier than at time p_R/α. Moreover, if the server is slowed down (i.e., moving at speed α) at the time of serving r_R, it is served precisely at time p_R/α.*

3.2 Upper bound for the competitive ratio of LeftGuard

We now turn to analyzing the algorithm LeftGuard. More precisely, we show the following.

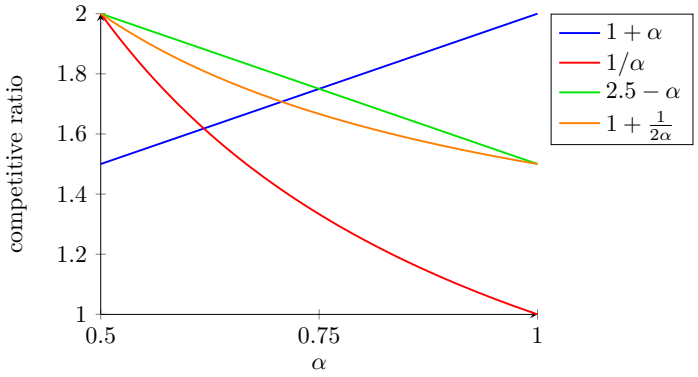

Figure 1 Bounds on the competitive ratio from Theorem 8 depending on α.

▶ **Theorem 8.** *For $\alpha \in (0.3, 1]$, the competitive ratio of* LeftGuard(α) *is upper-bounded by* $\max\{1+\alpha, 1/\alpha, 2.5-\alpha, 1+1/(2\alpha)\}$.

Note that, by setting $\alpha = 0.75$, we obtain a competitive ratio of 1.75 (cf. Figure 1), i.e., the statement of the first part of Theorem 2 follows.

We now prove Theorem 8. To this end, for the remainder of this subsection, we fix an arbitrary input sequence of requests, and we show that LeftGuard(α) achieves the claimed ratio on this sequence. We denote by Opt an optimum offline solution, or its completion time, depending on context. We denote the last right-request served by LeftGuard(α) by $r_R = (t_R, p_R)$, and the last left-request served by LeftGuard(α) by $r_L = (t_L, p_L)$. In case there is no left-request, it is immediate that the competitive ratio is bounded by $1/\alpha$ (by Observations 5 and 7 iii), and in case there is no right-request, it is bounded by $1+\alpha$ (by Observations 5 and 7 ii). Similarly, we immediately obtain the desired bounds in the following two cases.

▶ **Observation 9.** *If one of the following conditions hold, then* LeftGuard(α) *completes the request sequence by* $\max(1+\alpha, 1/\alpha)$Opt*:*

i) LeftGuard(α) *serves r_L after r_R.*

ii) LeftGuard(α) *serves r_R while moving at speed α*

Therefore, we assume from now on that r_L and r_R both exist, and that the server served first r_L and then moved at full speed to r_R. Next, we identify two further simple cases within these assumptions.

▶ **Lemma 10.** *Assume that* LeftGuard(α) *serves r_L before r_R and always moves at unit speed after serving r_L. If one of the following two conditions hold, the request sequence is completed by time* $(1+\alpha)$Opt.

i) Opt *serves r_L before r_R.*

ii) $t_L \le t_R$.

Proof. In both cases, Opt serves neither of r_L, r_R earlier than at time t_L, and then travels to the other request, which takes $p_R - p_L$ time units. Therefore,

$$\text{Opt} \ge t_L + p_R - p_L\,. \tag{4}$$

By assumption, LeftGuard first serves r_L and then moves at full speed to p_R. By Observation 7 ii), r_L is reached no later than $(1+\alpha)t_L$ and then moving to r_R takes $p_R - p_L$ time units. It follows that the request sequence is completed by time

$$(1+\alpha)t_L + p_R - p_L = t_L + p_R - p_L + \alpha t_L \overset{(4)}{\le} \text{Opt} + \alpha t_L \overset{\text{Obs. 5}}{\le} (1+\alpha)\text{Opt}\,. \qquad \blacktriangleleft$$

The remaining case turns out to be the most difficult to analyze and constitutes the crux of our proof. While the two previous results hold for any $\alpha \in (0, 1]$, we need in the following lemma that $\alpha \geq 0.3$. This is a reasonable assumption, since for $\alpha \leq 0.3$, the bound $1/\alpha$ from Observation 9 is already signigicantly worse than the bound of 2 achieved by LeftFirst.

▶ **Lemma 11.** *Assume that* LeftGuard(α) *serves* r_L *before* r_R *and always moves at unit speed after serving* r_L*. Additionally, assume that* $t_L > t_R$ *and* Opt *serves* r_R *before* r_L*. Then* LeftGuard(α) *completes the request sequence by time* $\max(2.5 - \alpha, 1 + 1/(2\alpha))\textsc{Opt}$.

Proof. First, observe that in the given setting, when r_L appeared at time t_L, the server decided to not detour to serve r_R before r_L. In other words, at time t_L, the condition in line 2 of Algorithm 2 was violated, i.e., we have

$$\max\left\{t_L + p_R - \text{pos}(t_L), \frac{p_R}{\alpha}\right\} + (p_R - p_L) > (1 + \alpha)t_L\,. \tag{5}$$

Further, note that the completion time of LeftGuard(α) is given by

$$\textsc{LeftGuard}(\alpha) \leq t_L + (\text{pos}(t_L) - p_L) + (p_R - p_L) = t_L + \text{pos}(t_L) + p_R - 2p_L \tag{6}$$

and the completion time of the offline optimum is bounded by

$$\textsc{Opt} \geq t_R + p_R - p_L. \tag{7}$$

Next, we will make use of the following observation: The behavior of LeftGuard is independent of multiplying all release times and positions with the same factor $x > 0$. In particular, this allows us to assume w.l.o.g. that $\textsc{Opt} = 1$ by rescaling with $1/\textsc{Opt}$ (if $\textsc{Opt} = 0$, then $\textsc{Alg} = 0$ is trivial). Summarizing our findings, it follows that the competitive ratio is bounded by the maximum value of the following optimization problem (here, Alg, Opt, $t_L, t_R, p_L, p_L, \text{pos}(t_L)$ are interpreted as variables of the optimization problem):

$$\begin{aligned}
\max\quad & \textsc{Alg} && \\
\text{s.t.}\quad & \textsc{Opt} \geq t_L && \text{(Observation 5)}\\
& \textsc{Opt} \geq t_R + p_R - p_L && (7)\\
& \textsc{Alg} \leq t_L + \text{pos}(t_L) + p_R - 2p_L && (6)\\
& \max(t_L + p_R - \text{pos}(t_L), p_R/\alpha) + p_R - p_L \geq (1 + \alpha)t_L && \text{(5, relaxed)}\\
& \text{pos}(t_L) \leq p_R && (r_R \text{ is a right-request})\\
& t_R \geq p_R && \text{(by assumption)}\\
& \textsc{Opt} = 1 && \text{(scaling)}\\
& \textsc{Alg}, \textsc{Opt}, t_R, t_L, p_R, p_L, \text{pos}(t_L) \geq 0\,. &&
\end{aligned}$$

Observe that the problem is not linear due to the maximum-operation in (5). Therefore, we distinguish now the two cases $t_L + p_R - \text{pos}(t_L) \geq p_R/\alpha$ and $t_L + p_R - \text{pos}(t_L) < p_R/\alpha$. For each of the two cases, we obtain a linear program, which allows us to use duality. In the end, the competitive ratio will be bounded by the maximum of the two values of the optimization problems.

First, we consider the case $t_L + p_R - \text{pos}(t_L) \geq p_R/\alpha$. Here, we obtain the following LP, where it turns out that the condition $\text{pos}(t_L) \leq p_R$ can be dropped.

$$
\begin{aligned}
\max\quad & \textsc{Alg} \\
\text{s.t.}\quad & \textsc{Opt} \geq t_L && \text{(Observation 5)} \\
& \textsc{Opt} \geq t_R + p_R - p_L && (7) \\
& \textsc{Alg} \leq t_L + \mathrm{pos}(t_L) + p_R - 2p_L && (6) \\
& t_L + p_R - \mathrm{pos}(t_L) + p_R - p_L \geq (1+\alpha)t_L && \text{(5, relaxed)} \\
& t_R \geq p_R && \text{(by assumption)} \\
& \textsc{Opt} = 1 && \text{(scaling)} \\
& \textsc{Alg}, \textsc{Opt}, t_R, t_L, p_R, p_L, \mathrm{pos}(t_L) \geq 0\,.
\end{aligned}
$$

A relaxation of the LP and its dual (D) can equivalently be written as

$$
\text{(P)} \qquad \max \quad (1,0,0,0,0,0,0)^\top \boldsymbol{x} \quad \text{s.t.}
$$

$$
\begin{pmatrix}
0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 1 & -1 & 0 \\
1 & 0 & 0 & -1 & -1 & 2 & -1 \\
0 & 0 & 0 & \alpha & -2 & 1 & 1 \\
0 & 0 & -1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\boldsymbol{x} \leq
\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
$$

$$
\boldsymbol{x} \geq \mathbf{0}
$$

$$
\text{(D)} \qquad \min \quad (0,0,0,0,0,1)^\top \boldsymbol{y} \quad \text{s.t.}
$$

$$
\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & -1 & 0 \\
1 & 0 & -1 & \alpha & 0 & 0 \\
0 & 1 & -1 & -2 & 1 & 0 \\
0 & -1 & 2 & 1 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0
\end{pmatrix}
\boldsymbol{y} \geq
\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}
$$

$$
\boldsymbol{y} \geq \mathbf{0}\,.
$$

By weak duality, any feasible solution to the dual provides an upper bound on the optimum value of the primal. The dual solution $\boldsymbol{y} = (1-\alpha, 1.5, 1, 1, 1.5, 2.5-\alpha)^\top$ is feasible for $\alpha \in (0,1]$ and has objective value $2.5 - \alpha$.

Next, we consider the case $t_L + p_R - \mathrm{pos}(t_L) < p_R/\alpha$. Here, we obtain the following LP, where it turns out that the condition $\textsc{Opt} \geq t_L$ can be dropped.

$$
\begin{aligned}
\max\quad & \textsc{Alg} \\
\text{s.t.}\quad & \textsc{Opt} \geq t_R + p_R - p_L && (7) \\
& \textsc{Alg} \leq t_L + \mathrm{pos}(t_L) + p_R - 2p_L && (6) \\
& \frac{1}{\alpha} p_R + p_R - p_L \geq (1+\alpha)t_L && \text{(5, relaxed)} \\
& \mathrm{pos}(t_L) \leq p_R && (r_R \text{ is a right-request}) \\
& t_R \geq p_R && \text{(by assumption)} \\
& \textsc{Opt} = 1 && \text{(scaling)} \\
& \textsc{Alg}, \textsc{Opt}, t_R, t_L, p_R, p_L, \mathrm{pos}(t_L) \geq 0\,.
\end{aligned}
$$

As before, we can write the dual (D) of the relaxed LP as

$$\text{(D)} \quad \begin{array}{l} \min \quad (0,0,0,0,0,1)^\top \boldsymbol{y} \quad \text{s.t.} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 1+\alpha & 0 & 0 & 0 \\ 1 & -1 & -\frac{\alpha+1}{\alpha} & -1 & 1 & 0 \\ -1 & 2 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \end{pmatrix} \boldsymbol{y} \geq \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\ \boldsymbol{y} \geq \boldsymbol{0}\,. \end{array}$$

The dual solution $\boldsymbol{y} = (1 + \frac{1}{2\alpha}, 1, \frac{1}{1+\alpha}, 1, 1 + \frac{1}{2\alpha}, 1 + \frac{1}{2\alpha})^\top$ is feasible for $\alpha \geq 0.3$ (More precisely, the second-to-last inequality holds for $\alpha \geq (\sqrt{17} - 3)/4 \approx 0.281$. All other inequalities hold for all $\alpha > 0$.) The corresponding objective value is $1 + \frac{1}{2\alpha}$.

Together, we obtain that the competitive ratio is bounded by $\max(2.5-\alpha, 1+1/(2\alpha))$. ◀

Let us summarize the different cases we considered to prove Theorem 8.

Proof of Theorem 8. First, we noted that, in case there is no left-request or no right-request, we easily obtain a bound of $\max(1 + \alpha, 1/\alpha)$ on the competitive ratio. Therefore, we can assume that a left- and right-request exist and let r_L denote the last-served left-request and r_R the last-served right-request. By Observation 9, we can assume that the server first served r_L and then moved at full-speed to r_R (otherwise, we obtain a bound of $\max(1 + \alpha, 1/\alpha)$). By Lemma 10, we can additionally assume that Opt serves r_R before r_L and that r_R was released before r_L (otherwise, we obtain a bound of $1 + \alpha$). Finally, we showed in Lemma 11 that, under these assumptions, the competitive ratio is bounded by $\max(2.5 - \alpha, 1 + 1/(2\alpha))$. ◀

3.3 Lower bound for the competitive ratio of LeftGuard

In this section, we present two lower-bound constructions for LeftGuard, showing that the analysis in Section 3.2 is tight in the sense that no choice of α yields a competitive ratio below 1.75 for LeftGuard(α) (i.e., we prove the second part of Theorem 2).

▶ **Lemma 12.** *The competitive ratio of* LeftGuard(α) *is at least* $\max(1 + \alpha, 1/\alpha)$ *for every* $\alpha \in (0, 1]$.

Proof. First, consider the input sequence consisting of the single request $(1, 0)$. Note that LeftGuard(α) executes MoveRight(α) before time 1 so that $\text{pos}(1) = \alpha$. Therefore, the request is served by time $1 + \alpha$. On the other hand, Opt serves the request immediately at time 1 by waiting in the origin. Therefore, the competitive ratio is at least $1 + \alpha$.

Next, consider the input sequence consisting of the single request $(1, 1)$. Since $(1, 1)$ is a right-request, by Observation 7 iii), LeftGuard(α) completes it not earlier than at time $1/\alpha$, while we have $\text{Opt} = 1$. Therefore, the competitive ratio is at least $1/\alpha$. ◀

▶ **Lemma 13.** *The competitive ratio of* LeftGuard(α) *is at least* $2.5 - \alpha$ *for* $\alpha \in [0.5, 1]$.

Proof. Consider the input sequence consisting of the three requests $(\frac{\alpha}{1+\alpha}, 0)$, $(0.5+\varepsilon, 0.5+\varepsilon)$ and $(1, 0)$ where $0 < \varepsilon < 0.5$. Observe that $\alpha/(1 + \alpha) \leq 0.5$ for $\alpha \leq 1$ so the requests indeed arrive in the order given above.

Opt can serve the requests by moving to position $0.5+\varepsilon$ at time $0.5+\varepsilon$ and then moving to position 0, i.e., we have

$$\text{Opt} = 1 + 2\varepsilon. \tag{8}$$

On the other hand, LeftGuard(α) moves away from the origin with speed α until time $\alpha/(1+\alpha)$ so that $\text{pos}(\alpha/(1+\alpha)) = \alpha^2/(1+\alpha)$. By moving back to the origin at unit speed, LeftGuard(α) could serve the first request at time

$$\frac{\alpha}{1+\alpha} + \frac{\alpha^2}{1+\alpha} = \alpha = (1+\alpha) \cdot \frac{\alpha}{1+\alpha}\,,$$

which is the maximum time allowed to serve the first request according to the algorithm definition. Therefore, LeftGuard(α) cannot decide to serve another right-request before serving the left-request at position 0, or before another left-request is revealed. Hence, LeftGuard(α) serves the left-request at position 0 at time $\alpha \leq 1$, and then moves right at maximum speed maintaining $\text{pos}(t) \leq \alpha t$ to serve the right-request at position $0.5+\varepsilon$. We obtain $\text{pos}(1) = 1-\alpha < 0.5+\varepsilon$ (where we have used that $\alpha \geq 0.5$), i.e., the request $(0.5+\varepsilon, 0.5+\varepsilon)$ is still unserved when the new left-request $(1,0)$ arrives.

Serving the right-request first, Alg would arrive at the left-request at time

$$1 + (0.5+\varepsilon) - \text{pos}(1) + 0.5 + \varepsilon = 2 + 2\varepsilon - (1-\alpha) = 1 + \alpha + 2\varepsilon > (1+\alpha) \cdot 1\,,$$

hence LeftGuard(α) has to move to the left-request first before serving the right-request. It follows that LeftGuard(α) finishes at time

$$1 + \text{pos}(1) + 0.5 + \varepsilon = 1 + (1-\alpha) + 0.5 + \varepsilon = 2.5 - \alpha + \varepsilon\,.$$

Together with (8) and letting $\varepsilon \to 0$, we obtain that the competitive ratio is at least $2.5-\alpha$. ◀

For $\alpha \leq 0.5$, Lemma 12 implies that the competitive ratio of LeftGuard(α) is at least $1/\alpha \geq 2$. For $\alpha > 0.5$, Lemmas 12 and 13 together imply that the competitive ratio is at least $\max(1+\alpha, 2.5-\alpha) \geq 1.75$. Thus, this completes the proof that LeftGuard(α) has competitive ratio at least 1.75 for any choice of α (i.e., the second part of Theorem 2).

4 General lower bound for online TSP on the half-line

In the following, we fix an arbitrary online algorithm Alg and prove a general lower bound for the competitive ratio of online TSP on the half-line.

▶ **Theorem 3.** *Every algorithm for open online TSP on the half-line has competitive ratio at least 1.6463.*

It is immediate to see that no online algorithm for the TSP on the half-line can achieve a competitive ratio strictly smaller than 3/2: At time 1, we reveal either a request at 0 if $\text{pos}(1) \geq 1/2$, or a request at 1 otherwise. In both cases, we have Alg $\geq 3/2$ and Opt $= 1$.

For the remainder of this section, we fix $\rho \in (3/2, 5/3]$ and assume that Alg has competitive ratio strictly less than ρ. We progressively introduce requests that, based on the fact that it must be ρ-competitive if no further requests appear, constrain Alg's behavior more and more. Specifically, we present two fixed requests, followed by three requests that depend on Alg's behavior. Based on the derived restrictions on Alg, we show that the resulting request sequence yields a lower bound of $\rho \geq 1.6463$. We denote by $\text{serve}(r_i)$ the time when the request $r_i = (t_i, p_i)$ is served by Alg and formally define the adversarial request sequence as follows.

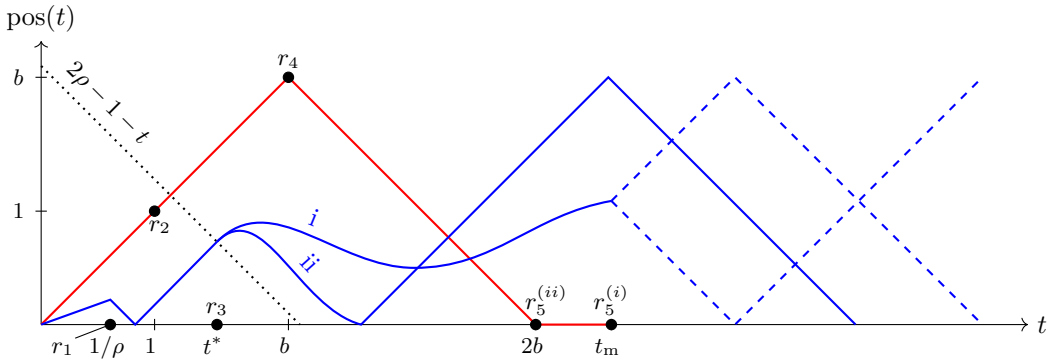

Figure 2 Adversarial request sequence for the lower bound in Theorem 3 for $\rho = 1.64$ with best ALG in blue and the offline optimum in red. Curved lines in ALG's trajectory indicate that the online algorithm has degrees of freedom in its behavior, whereas straight lines indicate that any other behavior increases the lower bound on the competitive ratio. Dashed lines show that there are two possible trajectories. The trajectories marked (i) and (ii) correspond to the two cases in the definition of r_5.

▶ **Definition 14** (Adversarial request sequence). *We present an adversarial request sequence* $(r_1, r_2, r_3, r_4, r_5)$ *consisting of the requests*

1. $r_1 := (1/\rho, 0)$ *and* $r_2 := (1, 1)$,
2. $r_3 := (t^*, 0)$, *where* $t^* := \min\{t \geq 1 : \mathrm{pos}(t) = 2\rho - 1 - t\}$,
3. $r_4 := (b, b)$ *for some* $b \geq t^* + 1/2$ *to be determined later,*
4. r_5 *depending on* ALG*'s behavior:*
   - **i.** *if* $\mathrm{pos}(2b) < b/2$ *and* $\mathrm{serve}(r_4) > 2b$, *then* $r_5 := (t_\mathrm{m}, 0)$ *with* $t_\mathrm{m} := \min\{t \geq 2b : \mathrm{pos}(t_\mathrm{m}) = b/2\}$,
   - **ii.** *otherwise,* $r_5 := (2b, 0)$.

We show in Lemmas 15 and 18 that this sequence is well-defined, in the sense that the time t^* in the second step and, if needed, the time t_m in the fourth step exist.

The adversarial sequence is illustrated in Figure 2 for $\rho = 1.64$. The figure also shows the trajectory of the optimum offline solution for the entire request sequence, as well as the best possible trajectory of ALG.

By $\sigma_{\leq i} := (r_1, r_2, \ldots, r_i)$ we denote the sequence of requests revealed up to and including request r_i, and we write $\mathrm{ALG}(\sigma_{\leq i})$ and $\mathrm{OPT}(\sigma_{\leq i})$ for the cost of ALG and the offline optimum, respectively, for serving the request sequence $\sigma_{\leq i}$. Moreover, we write $\mathrm{ALG}[\sigma_{\leq i}]$ for the schedule ALG computes for the request sequence $\sigma_{\leq i}$, i.e., ALG's behavior if only the requests $r_1, \ldots, r_i$ were revealed. Note that the serving times $\mathrm{serve}(r_i)$ are always w.r.t. the full schedule $\mathrm{ALG}[\sigma_{\leq 5}]$.

The following lemma establishes that the time t^* as described in the second step of Definition 14 exists and falls within certain bounds.

▶ **Lemma 15.** *After releasing the requests* r_1 *and* r_2 *of Definition 14, there exists a time* $t^* \in [\rho + (3/(2\rho)) - 1, 2]$ *such that* $\mathrm{pos}(t^*) = 2\rho - 1 - t^*$. *Moreover,* $\mathrm{serve}(r_2) > t^*$.

Proof. Observe that, before the first request r_1 is revealed, the server needs to be able to handle the case where the first request is of the form (t, t) with $t \leq 1/\rho$. In this case, the offline optimum's cost is t and the server must be able to serve the request before time ρt. Note that $\mathrm{pos}(t) \leq t$ since the server is restricted to unit speed. It follows that $t + (t - \mathrm{pos}(t)) \leq \rho t$.

Equivalently, we have

$$\mathrm{pos}(t) \geq (2-\rho)t \quad \text{for all } t \in [0, 1/\rho].$$

This implies that, when r_1 is revealed at time $1/\rho$, we have $\mathrm{pos}(1/\rho) \geq (2-\rho)/\rho$, and it follows that

$$\mathrm{serve}(r_1) \geq t_1 + |\mathrm{pos}(t_1) - p_1| = 1/\rho + \mathrm{pos}(1/\rho) \geq (3-\rho)/\rho. \tag{9}$$

Moreover, $\mathrm{serve}(r_1) \leq 1$ because ALG is ρ-competitive for the request sequence $\sigma_{\leq 1}$, i.e., r_1 is already served when r_2 is revealed. From (9), it follows that

$$\mathrm{serve}(r_2) \geq \mathrm{serve}(r_1) + |p_2 - \mathrm{pos}(\mathrm{serve}(r_1))| = \mathrm{serve}(r_1) + 1 \geq (3-\rho)/\rho + 1 = 3/\rho.$$

With this, we obtain that there is a time t^* where $\mathrm{pos}(t^*) = 2\rho - 1 - t^*$: Let $\ell(t) := 2\rho - 1 - t$. At time $\mathrm{serve}(r_1)$, we have $\mathrm{pos}(\mathrm{serve}(r_1)) = 0 < \ell(\mathrm{serve}(r_1))$. When request $r_2 = (1,1)$ is served at time $\mathrm{serve}(r_2) \geq 3/\rho$, we have $\mathrm{pos}(\mathrm{serve}(r_2)) = 1 > 2\rho - 1 - (3/\rho) \geq \ell(\mathrm{serve}(r_2))$, where the first inequality holds for $\rho \leq 1.82$. Therefore, by continuity of ℓ, there exists $t^* \in (\mathrm{serve}(r_1), \mathrm{serve}(r_2))$ such that $\mathrm{pos}(t^*) = \ell(t^*) = 2\rho - 1 - t^*$.

It remains to prove the claimed upper and lower bounds on t^*. To this end, note that r_2 is served no earlier than time $t^* + (1 - \mathrm{pos}(t^*)) = t^* + 1 - (2\rho - 1 - t^*) = 2t^* + 2 - 2\rho$. Moreover, $\mathrm{OPT}(\sigma_{\leq 2}) \leq 1 + 1/\rho$ (wait in $p_1 = 0$ until time $t_1 = 1/\rho$ and then move up to $p_2 = 1$) so that $2t^* + 2 - 2\rho \leq \rho\mathrm{OPT}(\sigma_{\leq 2}) \leq \rho + 1$. Reformulating and using $\rho \leq 5/3$ yields

$$t^* \leq \frac{3\rho - 1}{2} \leq 2\,.$$

To bound t^* from below, observe that, using (9) and $\mathrm{pos}(\mathrm{serve}(r_1)) = p_1 = 0$, we have for all $t \geq \mathrm{serve}(r_1)$ that $\mathrm{pos}(t) \leq \mathrm{pos}(\mathrm{serve}(r_1)) + t - \mathrm{serve}(r_1) \leq t - (3-\rho)/\rho$. By definition of time t^*, it therefore holds that

$$2\rho - 1 - t^* = \mathrm{pos}(t^*) \leq t^* - (3-\rho)/\rho\,,$$

or, equivalently, $t^* \geq \rho + 3/(2\rho) - 1$. ◀

The time t^* is chosen precisely such that ALG is forced to serve r_2 before r_3, as the alternative would not be competitive in case no more requests are released.

▶ **Observation 16** (Planned order of r_2 and r_3)**.** $\mathrm{ALG}[\sigma_{\leq 3}]$ *serves* $r_2 = (1,1)$ *before* $r_3 = (t^*, 0)$.

Proof. Otherwise, $\mathrm{ALG}[\sigma_{\leq 3}]$ would finish the request sequence no earlier than

$$t^* + (\mathrm{pos}(t^*) - 0) + 1 = t^* + 2\rho - 1 - t^* + 1 = 2\rho \overset{\text{Obs. 17}}{\geq} \rho\mathrm{OPT}(\sigma_{\leq 3})\,,$$

contradicting that ALG has competitive ratio strictly less than ρ. ◀

To prove restrictions on ALG's behavior beyond this, we use the fact that ALG's competitive ratio is strictly less than ρ for the request sequences $\sigma_{\leq 3}$ and $\sigma_{\leq 4}$, whose optimum offline cost can be bounded.

▶ **Observation 17.** *It holds that*

$$\mathrm{OPT}(\sigma_{\leq 3}) \leq 2 \quad \textit{and} \quad \mathrm{OPT}(\sigma_{\leq 4}) \leq t^* + b.$$

Proof. An offline algorithm can serve $\sigma_{\leq 3}$ by moving to $p_2 = 1$ at time $t_2 = 1$ and then moving back to $p_1 = p_3 = 0$ to serve $r_1 = (1/\rho, 0)$ and $r_3 = (t^*, 0)$ at time 2, since $t^* \leq 2$, achieving completion time 2.

For $\sigma_{\leq 4}$, the offline optimum can wait in 0 until time t^* and then move to $p_4 = b > 1$, serving r_2 on the way. ◀

We can now prove lower bounds on the competitive ratio of ALG, depending on which request r_5 is released.

▶ **Lemma 18.** *If* $\mathrm{pos}(2b) < b/2$ *and* $\mathrm{serve}(r_4) > 2b$*, then the adversarial request sequence is well-defined in the sense that* t_m *as described in Definition 14 exists, and*

$$\rho \geq 1 + \frac{3b/2}{\rho(t^* + b) - b/2}.$$

Proof. Since $\mathrm{pos}(2b) < b/2$ and $r_4 = (b, b)$ is not yet served at time $2b$, the server has to cross position $b/2$ at some time t_m after $2b$. Therefore, the request $r_5 = (t_\mathrm{m}, 0)$ is well-defined.

Between times b and t_m, only requests up to r_4 are present, so the offline optimum cost is $\mathrm{OPT}(\sigma_{\leq 4}) \leq t^* + b$ by Observation 17. Since ALG is ρ-competitive, $\mathrm{ALG}[\sigma_{\leq 4}]$ must therefore serve $r_4 = (b, b)$ before time $\rho(t^* + b)$. This implies that

$$t_\mathrm{m} + b/2 = t_\mathrm{m} + |p_4 - \mathrm{pos}(t_\mathrm{m})| \leq \rho(t^* + b)\,. \tag{10}$$

Since $\mathrm{pos}(t_\mathrm{m}) = b/2$ and since $r_4 = (b, b)$ and $r_5 = (t_\mathrm{m}, 0)$ are not yet served by ALG at time t_m, we have $\mathrm{ALG}(\sigma_{\leq 5}) \geq t_\mathrm{m} + 3b/2$. Moreover, $\mathrm{OPT}(\sigma_{\leq 5}) = t_\mathrm{m}$. It follows that

$$\rho \geq \frac{\mathrm{ALG}}{\mathrm{OPT}} \geq \frac{t_\mathrm{m} + 3b/2}{t_\mathrm{m}} = 1 + \frac{3b/2}{t_\mathrm{m}} \overset{(10)}{\geq} 1 + \frac{3b/2}{\rho(t^* + b) - b/2}\,.$$ ◀

To prove a lower bound on the competitive ratio in the second case, we first establish that ALG must plan to serve r_3 before r_4, and we derive a lower bound on the time r_3 is served. We prove the claimed serving order by contradiction: assuming the opposite, we can construct an additional request for which the resulting request sequence even yields a lower bound of 1.75. To establish the lower bound on the serving time of r_3, we make use of the fact that ALG must plan to serve r_2 before r_3 (Observation 16). The details of both proofs can be found in the full version of this paper.

▶ **Lemma 19** (Serving order of r_3 and r_4)**.** $\mathrm{ALG}[\sigma_{\leq 4}]$ *serves* $r_3 = (t^*, 0)$ *before* $r_4 = (b, b)$*.*

▶ **Lemma 20** (Serving time of r_3)**.** *If* $b \geq t^* + 1/2$*, it holds that* $\mathrm{serve}(r_3) \geq 3 + 2t^* - 2\rho$*.*

Combining the previous two lemmas, we can prove a lower bound on the competitive ratio for the second case. Here, the main observation is that it is always optimal for ALG to serve r_3, then r_4 and then r_5, which allows us to use the lower bound on $\mathrm{serve}(r_3)$ to obtain a lower bound on the competitive ratio for $\sigma_{\leq 5}$.

▶ **Lemma 21.** *Let* $b \geq t^* + 1/2$*. If* $\mathrm{pos}(2b) \geq b/2$ *or* $\mathrm{serve}(r_4) \leq 2b$*, then*

$$\rho \geq 1 + \frac{3 + 2t^* - 2\rho}{2b}\,.$$

Proof. The adversarial sequence concludes by releasing $r_5 = (2b, 0)$ at time $2b$. The optimum offline cost for the entire sequence is $\mathrm{OPT}(\sigma_{\leq 5}) = 2b$ (immediately move to b and back to the origin in time $2b$). By Lemma 19, $\mathrm{ALG}[\sigma_{\leq 4}]$ serves $r_3 = (t^*, 0)$ before $r_4 = (b, b)$.

If $\text{serve}(r_4) \leq 2b = t_5$, i.e., r_4 has already been served when r_5 appears, then we have $\text{serve}(r_3) < \text{serve}(r_4) < \text{serve}(r_5)$. It follows that $\textsc{Alg}(\sigma_{\leq 5}) \geq \text{serve}(r_3) + |p_4 - p_3| + |p_5 - p_4| = \text{serve}(r_3) + 2b$. In the other case (i.e., $\text{serve}(r_4) > 2b$), we have $\text{pos}(2b) \geq b/2$ and $\text{serve}(r_4) > 2b$ by assumption. Moreover, we can show that $\text{serve}(r_3) < 2b$, i.e., r_3 is served before r_4 and r_5: Otherwise, since $\textsc{Alg}[\sigma_{\leq 4}]$ serves r_3 before r_4 (Lemma 19), it holds that

$$2\rho b \overset{b \geq t^*}{\geq} \rho(t^* + b) \overset{\text{Obs. } 17}{\geq} \rho\textsc{Opt}(\sigma_{\leq 4}) \geq \textsc{Alg}(\sigma_{\leq 4}) \geq 2b + |\text{pos}(2b) - p_3| + |p_4 - p_3| \geq 7b/2\,,$$

contradicting $\rho \leq 5/3$. Since $\text{pos}(2b) \geq b/2$ and $\text{serve}(r_4) > 2b$, the fastest way for $\textsc{Alg}$ to complete the sequence $\sigma_{\leq 5}$ is to serve $r_4 = (b, b)$ before $r_5 = (2b, 0)$. As before, we obtain $\textsc{Alg}(\sigma_{\leq 5}) \geq \text{serve}(r_3) + 2b$. It follows that

$$\rho \geq \frac{\textsc{Alg}(\sigma_{\leq 5})}{\textsc{Opt}(\sigma_{\leq 5})} \geq \frac{\text{serve}(r_3) + 2b}{2b} = 1 + \frac{\text{serve}(r_3)}{2b} \overset{\text{Lem. } 20}{\geq} 1 + \frac{3 + 2t^* - 2\rho}{2b}\,. \qquad \blacktriangleleft$$

To prove Theorem 3, we also need the following two technical lemmas, which are proven in the full version of this paper.

▶ **Lemma 22.** *Let $\rho \in [3/2, 5/3]$ and $t^* \geq \frac{7}{5}$. If b satisfies*

$$\frac{3 + 2t^* - 2\rho}{2b} = \frac{3b/2}{\rho(t^* + b) - b/2} \tag{11}$$

then $b \geq t^* + 1/2$.

▶ **Lemma 23.** *Assume that $\rho > 3/2$ is fixed and $b(t)$ is chosen such that*

$$\frac{3 + 2t - 2\rho}{2b(t)} = \frac{3b(t)}{2\rho(t + b(t)) - b(t)} =: \text{LB}(t).$$

Then $\text{LB}(t)$ is increasing in t.

Proof of Theorem 3. Lemmas 18 and 21 combined yield that, for every $b \geq t^* + 1/2$ and $\rho \in (3/2, 5/3]$, we have

$$\rho \geq 1 + \min\left(\frac{3b/2}{\rho(t^* + b) - b/2}, \frac{3 + 2t^* - 2\rho}{2b}\right) =: \text{LB}(t^*, \rho, b). \tag{12}$$

The remainder of the proof is purely analytical and dedicated to showing that, for every $\rho \in (3/2, 5/3]$ and $t^* \geq \rho + 3/(2\rho) - 1$, there exists $b \geq t^* + 1/2$ such that, in (12) one of the values in the minimum is at least 1.64634.

To this end, we choose the maximal $b(t^*, \rho)$ such that the two values in the minimum coincide. By Lemma 15, we have that $t^* \geq \rho + \frac{3}{2\rho} - 1 \geq \frac{3}{2} + \frac{3}{2} \cdot \frac{3}{5} - 1 = 7/5$, and therefore, Lemma 22 gives $b(t^*, \rho) \geq t^* + 1/2$, so that the lower bound still applies. Moreover, it follows from Lemma 23 that the resulting lower bound $\text{LB}(t^*, \rho, b(t^*, \rho))$ is increasing in t^*. Therefore, we obtain a general lower bound by setting $t^* = \rho + 3/(2\rho) - 1$ (since we have proven in Lemma 15 that t^* is lower bounded by this value). Plugging this in, we obtain

$$\rho \geq 1 + \min\left(\frac{3b}{2\rho^2 - 2\rho + 2\rho b + 3 - b}, \frac{3 + \rho}{2\rho b}\right)$$

resulting in

$$b = \frac{2\rho^2 + 5\rho - 3 + \sqrt{52\rho^4 + 116\rho^3 - 59\rho^2 + 186\rho + 9}}{12\rho}$$

and a lower bound on the competitive ratio of

$$1 + \frac{6(\rho+3)}{2\rho^2+5\rho-3+\sqrt{52\rho^4+116\rho^3-59\rho^2+186\rho+9}},$$

which decreases in $\rho > 0$ and equals ρ at $\rho > 1.6463$. Therefore, we established a lower bound of at least 1.6463 for every $\rho \leq 1.6463$. ◀

References

1 Mikkel Abrahamsen, Ioana O. Bercea, Lorenzo Beretta, Jonas Klausen, and László Kozma. Online sorting and online TSP: randomized, stochastic, and high-dimensional. In *Proceedings of the 32nd Annual European Symposium on Algorithms (ESA)*, pages 5:1–5:15, 2024. doi:10.4230/LIPIcs.ESA.2024.5.

2 Norbert Ascheuer, Sven Oliver Krumke, and Jörg Rambau. Online dial-a-ride problems: Minimizing the completion time. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 639–650, 2000. doi:10.1007/3-540-46541-3_53.

3 Giorgio Ausiello, Vincenzo Bonifaci, and Luigi Laura. The on-line asymmetric traveling salesman problem. *Journal of Discrete Algorithms*, 6(2):290–298, 2008. doi:10.1016/J.JDA.2007.03.002.

4 Giorgio Ausiello, Marc Demange, Luigi Laura, and Vangelis Th. Paschos. Algorithms for the on-line quota traveling salesman problem. *Information Processing Letters*, 92(2):89–94, 2004. doi:10.1016/J.IPL.2004.06.013.

5 Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001. doi:10.1007/S004530010071.

6 Júlia Baligács, Yann Disser, Irene Heinrich, and Pascal Schweitzer. Exploration of graphs with excluded minors. *Journal of Computer and System Sciences*, 156:103725, 2026. doi:10.1016/J.JCSS.2025.103725.

7 Júlia Baligács, Yann Disser, Farehe Soheil, and David Weckbecker. Tight analysis of the lazy algorithm for open online dial-a-ride. In *Proceedings of the 18th International Algorithms and Data Structures Symposium (WADS)*, pages 43–64, 2023. doi:10.1007/978-3-031-38906-1_4.

8 Alexander Birx. *Competitive analysis of the online dial-a-ride problem*. PhD thesis, TU Darmstadt, 2020.

9 Alexander Birx and Yann Disser. Tight analysis of the smartstart algorithm for online dial-a-ride on the line. *SIAM Journal on Discrete Mathematics*, 34(2):1409–1443, 2020. doi:10.1137/19M1268513.

10 Alexander Birx, Yann Disser, and Kevin Schewior. Improved bounds for open online dial-a-ride on the line. *Algorithmica*, 2022. doi:10.1007/S00453-022-01061-4.

11 Antje Bjelde, Jan Hackfeld, Yann Disser, Christoph Hansknecht, Maarten Lipmann, Julie Meißner, Miriam Schlöter, Kevin Schewior, and Leen Stougie. Tight bounds for online TSP on the line. *ACM Transactions on Algorithms*, 17(1):3:1–3:58, 2021. doi:10.1145/3422362.

12 Michiel Blom, Sven O. Krumke, Willem E. de Paepe, and Leen Stougie. The online TSP against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001. doi:10.1287/IJOC.13.2.138.10517.

13 Vincenzo Bonifaci and Leen Stougie. Online k-server routing problems. *Theory of Computing Systems*, 45(3):470–485, 2008. doi:10.1007/S00224-008-9103-4.

14 George Christodoulou and Alkmini Sgouritsa. An improved upper bound for the universal TSP on the grid. *SIAM J. Comput.*, 53(5):1476–1523, 2024. doi:10.1137/17M1154849.

15 Romain Cosson. Breaking the $k/\log k$ barrier in collective tree exploration via tree-mining. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4264–4282, 2024. doi:10.1137/1.9781611977912.148.

16 Yann Disser, Jan Hackfeld, and Max Klimm. Tight bounds for undirected graph exploration with pebbles and multiple agents. *Journal of the ACM*, 66(6):40(41), 2019. doi:10.1145/3356883.

17 Esteban Feuerstein and Leen Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001. doi:10.1016/S0304-3975(00)00261-9.

18 Patrick Jaillet and Xin Lu. Online traveling salesman problems with service flexibility. *Networks*, 58(2):137–146, 2011. doi:10.1002/NET.20454.

19 Patrick Jaillet and Xin Lu. Online traveling salesman problems with rejection options. *Networks*, 64(2):84–95, 2014. doi:10.1002/NET.21559.

20 Vinay A. Jawgal, V. N. Muralidhara, and P. S. Srinivasan. Online travelling salesman problem on a circle. In *In Proceedings of the 15th International Conference on Theory and Applications of Models of Computation (TAMC)*, pages 325–336, 2019. doi:10.1007/978-3-030-14812-6_20.

21 Lujun Jia, Guolong Lin, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. Universal approximations for TSP, Steiner tree, and set cover. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 386–395, 2005. doi:10.1145/1060590.1060649.

22 Bala Kalyanasundaram and Kirk R. Pruhs. Constructing competitive tours from local information. *Theoretical Computer Science*, 130(1):125–138, 1994. doi:10.1016/0304-3975(94)90155-4.

23 Sven O. Krumke. Online optimization competitive analysis and beyond. Habilitation thesis, Zuse Institute Berlin, 2001.

24 Sven O. Krumke, Willem E. de Paepe, Diana Poensgen, Maarten Lipmann, Alberto Marchetti-Spaccamela, and Leen Stougie. 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)*, pages 258–269, 2006. doi:10.1007/11671411_20.

25 Sven Oliver Krumke, Luigi Laura, Maarten Lipmann, Alberto Marchetti-Spaccamela, Willem de Paepe, Diana Poensgen, and Leen Stougie. 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)*, pages 200–214, 2002. doi:10.1007/3-540-45753-4_18.

26 Maarten Lipmann. *On-line routing*. PhD thesis, Technische Universiteit Eindhoven, 2003.