TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Mathematics

Bachelor's Thesis

Abort-Strategies for Preemptive Online Dial-a-Ride

Roman Edenhofer

31st October, 2020

Supervisor: Prof. Dr. Yann Disser

Contents

Introduction **1**

1 Preliminaries **3**

2 State of the Art **5**

3 Algorithms **8**

3.1 Algorithm: ABORT 8

3.1.1 ABORT for Open Online Dial-a-Ride 9

3.1.2 ABORT for Closed Online Dial-a-Ride 10

3.2 Algorithm: ABORT-AND-WAIT 11

3.2.1 AAW for Open Online Dial-a-Ride 12

3.2.2 AAW for Closed Online Dial-a-Ride 15

3.3 Algorithm: ABORT-OR-REPLAN 17

3.3.1 AOR for Uncapacitated Closed Online Dial-a-Ride on the Halfline 17

Bibliography **34**

Thesis Statement **35**

Introduction

We consider the online Dial-a-Ride Problem where objects are to be transported between points in a metric space in the shortest possible completion time. In contrast to the offline case, these transportation requests arrive over time and can only be served after they are released. One main way to measure the quality of an online algorithm is via competitive analysis where we determine its *competitive ratio*, i.e., the smallest ratio we can guarantee for all possible requests between the completion time of the online algorithm and the completion time of the optimal (offline) solution which knows all requests in advance. Different competitive ratios are known for different variations of the problem.

This thesis is organized as follows: In the first chapter we formally define the problem with some of its variations and introduce notation. In the second chapter we give an overview of the state of the art of currently best known competitive ratios. In the third chapter, the main part of this thesis, we present three algorithms for the preemptive and uncapacitated open and closed case: ABORT, ABORT-AND-WAIT (AAW) and ABORT-OR-REPLAN (AOR). They are all based on the approach that whenever new requests arrive the server tries to abort its current schedule and return to its starting position to begin a new optimal schedule for the remaining requests from there. Table 1 gives an overview of the competitive ratios of the first strategy ABORT. The second strategy AAW extends ABORT by a waiting routine which improves its competitiveness. In Bjelde *et al.* [1] a similar algorithm has already been investigated for the open case on the real line. We slightly changed the algorithm such that it also works efficiently on general metric spaces for the open and closed case. Table 2 contains the competitive ratios of AAW. All of them

	ABORT	open	closed
general	uncapacitated (c= ∞)	3 [Thm 3.3]	2.5 [Thm 3.5]
	preemptive	3 [Thm 3.2]	2.5 [Thm 3.4]

Table 1: Competitive ratios of the ABORT-strategy.

	ABORT-AND-WAIT	open	closed
general	uncapacitated (c= ∞)	2.4142 [Thm 3.7]	2 [Thm 3.10]
	preemptive	2.4142 [Thm 3.6]	2 [Thm 3.9]

Table 2: Competitive ratios of the AAW-strategy.

match the currently best known upper bounds and the preemptive open case even improves the upper bound of 2.6180 from Lipmann [2, Thm 4.9]. The final algorithm we present is AOR for the uncapacitated closed case on the halfline. This algorithm further improves AAW in this setting. A current schedule is not always aborted but instead sometimes the server continues to serve unserved requests starting from its current position instead. For the closed case on the halfline this algorithm improves the best known upper bound of 2 given by Ascheuer *et al.* [3, Thm 6] for general metric spaces to $\big(3 + \sqrt{2}/2\big)/2 \approx 1.8536$ [Thm 3.12].

Acknowledgements

First of all, I want to thank my supervisor Prof. Dr. Yann Disser for entrusting me this fascinating topic. Furthermore, I am very grateful for his enriching literal recommendations that prepared my mind for new points of view and inspired me a lot. And, of course, I feel grateful towards my friends and family for all their helpful company in demanding times. In particular, I want to thank my parents and my brother for their thorough checking of the manuscript's spelling and their interest in the topic.

Chapter 1

Preliminaries

An instance of the basic online Dial-a-Ride problem consists of a metric space $M = (X, d)$ with a distinguished origin $0 \in X$ and a set of requests $\sigma = \{\sigma_1, ..., \sigma_n\}$. Each request is of the form $\sigma_i = (a_i, b_i; t_i)$ where $a_i, b_i \in M$ are the source and destination of the request, respectively, and $t_i \geq 0$ is the release time of the request. The requests are to be served by a single server starting at the origin in the shortest possible completion time, i.e., the time the last request is served.

We distinguish multiple versions of the problem. First of all, we give special attention to the case where source and destination of each request coincide. This case yields the traveling salesman problem (TSP). Here, we do not state source and destination separately for every request but instead write $(a_i; t_i)$ for requests in σ. We also look at different capacities $c \in \mathbb{N} \cup \{\infty\}$ which bound the number of requests the server can carry simultaneously. Here, we further distinguish the preemptive and non-preemptive case. The former means that requests can be unloaded at any point of time to be picked up and delivered to their destination at a later time. In contrast the latter only allows requests to be unloaded once at their destination. Lastly, we distinguish the closed and the open version. In the closed case the server must end its tour at the origin and in the open version the server can end at any position. Other versions of the problem have been investigated but are not treated in this thesis, e.g. multiple servers by Bonifaci *et al.* [4] or so called fair adversaries where the movement of the optimal solution is restricted by Blom *et al.* [5] and many more.

We use similar notation as introduced by Bjelde *et al.* [1] and Ascheuer *et al.* [3]:

- Let $\sigma_{\leq t}$ denote all unserved requests from σ released up to time t. Similarly, $\sigma_{>t}$ (or $\sigma_{\geq t}$) denotes all requests from σ released after time t (respectively at or after time t).

- Let OPT(σ) be the time it takes an optimal (offline) solution OPT which knows all requests in advance to serve every request from a set σ. Similarly for an online algorithm ALG, we denote by ALG(σ) the time it takes ALG to serve all requests from σ.

- Let a set of requests σ be fixed. Then $L(t, p, R)$ (respectively $L^*(t, p, R)$) denotes the length of a shortest possible open (respectively closed) schedule for $R \subseteq \sigma$ starting at time t and position p. Note that in the open case for all $0 \leq t \leq t', p, p' \in M, R \subseteq R' \subseteq \sigma$ we have

  $$L(t', p, R) \leq d(p, p') + L(t, p', R').$$

  The same property holds for $L^*(\cdot, \cdot, \cdot)$ in the closed case.

- For a given input, let $p^{\mathrm{ALG}}(t)$ denote the position of the ALG-server at time t. Similarly, let $p^{\mathrm{OPT}}(t)$ denote the position of the server of an optimal solution at time t.

- To specify the path of the server we write move(a) for a tour that moves from the current position to the point $a \in M$ in a shortest possible way and we write waituntil(t') for the tour that waits at the current position until time t'. Further the operator $\oplus$ is used to concatenate multiple tours. For a tour T, we write $|T|$ for its length.

We can now formally define what it means for an online algorithm ALG to be competitive.

Definition 1.1. An online algorithm ALG for Dial-a-Ride is said to be ρ-*competitive* if for all possible sets of requests σ we have

$$\mathrm{ALG}(\sigma) \leq \rho \mathrm{OPT}(\sigma).$$

The *competitive ratio* of ALG is the infimum over all $\rho \geq 1$ for which ALG is ρ-competitive.

Chapter 2

State of the Art

In recent works on online Dial-a-Ride the real line and halfline have received considerable attention. Table 2.1 gives an overview of the currently best known lower and upper bounds for competitive ratios of online Dial-a-Ride with special attention for these two metric spaces. This thesis will mainly focus on the uncapacitated and preemptive case. We also consider bounds for the non-preemptive case and the traveling salesmen problem as some of these carry over to the aformentioned cases.

For online TSP on the line, Bjelde *et al.* present conclusive results. For the closed case they present a 1.6404-competitive algorithm (see [1, Thm 3]) matching the lower bound given by Ausiello *et al.* (see [6, Thm 3.3]). For the open case they give a lower bound of 2.0346 (see [1, Thm 4]) and an algorithm matching that bound (see [1, Thm 10]). For online TSP on the halfline, conclusive bounds are only known for the closed case. Blom *et al.* show a lower bound of 1.5 (see [5, Thm 1]) and a matching algorithm (see [5, Thm 2]). For the open case a lower bound of 1.6272 is given by Lipmann (see [2, Thm 3.7]).

For non-preemptive Dial-a-Ride on the line and halfline currently unmachted lower bounds are known. For the closed case on the halfline a lower bound of 1.7071 is presented by Ascheuer *et al.* (see [3, Thm 2]) and on the line of 1.75 by Bjelde *et al.* (see [1, Thm 13]). For the open case on the halfline a lower bound of 1.8968 is given by Lipmann (see [2, Thm 3.14]) and on the line of 2.0585 by Birx *et al.* (see [7, Thm 1]). For non-preemptive Dial-a-Ride on general metric spaces, conclusive bounds are only known for the closed case. Here, Ascheuer *et al.* present the SMARTSTART-algorithm (see [3, Thm 6]) which matches the lower bound of 2 given by Ausiello *et al.* for closed online TSP on general metric spaces (see [6, Thm 3.2]). For the open case on general metric spaces the best known upper bound from Lipmann is 2.6180 (see [2, Thm 4.9]).

Conclusive results for preemptive or uncapacitated online Dial-a-Ride are also only known for the closed case. A close inspection of the proof of SMARTSTART's competitiveness from Ascheuer *et al.* shows that the upper bound of 2 carries over from the non-preemptive to the preemptive or uncapacitated case on general metric spaces. This closes the gap between lower and upper bound in these cases as the lower bound of 2 for closed online TSP from Ausiello *et al.* trivially carries over as well. A better upper bound than 2 on the halfline in the uncapacitated closed case is established in this thesis by AOR with 1.8536 (see [Thm 3.12]). For the preemptive and uncapacitated open case AAW establishes an upper bound of 2.4142 on general metric spaces (see [Thm 3.6,3.7]). No better upper bounds are known on the line or halfline. As mentioned earlier AAW is only a slight variation of an algorithm originally analyzed by Bjelde *et al.* for the open case on the line (see [1, Thm 12]). In a shortly to be published version they even show that their algorithm upholds the same competitive ratio for the uncapacitated open case on general metric spaces. Therefore AAW only improves the upper bound in the preemptive open case on general metric spaces. Here the former best known upper bound was 2.6180 by Lipmann which carried over from the non-preemptive case (see [2, Thm 4.9]). In a way this algorithm from Lipmann could even be seen as a natural transfer of AAW to the non-preemptive setting such that a schedule is only aborted if the non-preemption-condition allows to do so.

General Bounds		open		closed	
		lower bound	upper bound	lower bound	upper bound
general	non-preemptive ($c < \infty$)	2.0585	2.6180 [2, Thm 4.9]	2	2 [3, Thm 6]
general	uncapacitated ($c = \infty$)	2.0346	2.4142 [1, Thm 12]	2	2 [3, Thm 6]
general	preemptive	2.0346	$2.6180 \rightarrow$ **2.4142** [Thm 3.6]	2	2 [3, Thm 6]
general	TSP	2.0346	2.4142	2 [6, Thm 3.2]	2
line	non-preemptive ($c < \infty$)	2.0585 [7, Thm 1]	2.6180	1.75 [1, Thm 13]	2
line	uncapacitated ($c = \infty$)	2.0346	2.4142	1.6404	2
line	preemptive	2.0346	2.4142 [1, Thm 12]	1.6404	2
line	TSP	2.0346 [1, Thm 4]	2.0346 [1, Thm 10]	1.6404 [6, Thm 3.3]	1.6404 [1, Thm 3]
halfline	non-preemptive ($c < \infty$)	1.8968 [2, Thm 3.14]	2.6180	1.7071 [3, Thm 2]	2
halfline	uncapacitated ($c = \infty$)	1.6272	2.4142	1.5	$2 \rightarrow$ **1.8536** [Thm 3.12]
halfline	preemptive	1.6272	2.4142	1.5	2
halfline	TSP	1.6272 [2, Thm 3.7]	2.0346	1.5 [5, Thm 1]	1.5 [5, Thm 2]

Table 2.1: Overview of the best known bounds for the competitive ratios of online Dial-a-Ride. Bold results are new from this thesis. Results without reference are direct consequences from other bounds.

Chapter 3

Algorithms

3.1 Algorithm: ABORT

In this section we consider the ABORT-strategy (Algorithm 1): Whenever new requests are released the server returns to the origin while delivering currently carried requests or unloading them at their respective sources in the shortest possible way. Once the server arrives at the origin it starts a new optimal tour for all remaining requests from there.

We will find that in the preemptive (and uncapacitated) open case this algorithm is 3-competitive and in the corresponding closed case it is 2.5-competitive.

Algorithm 1: ABORT for Open/Closed Online Dial-a-Ride

```
this function is called upon receiving a new request
input  : unserved requests $\sigma_{\leq t}$, current server-position $p^{\mathrm{AAW}}(t)$
output: closed tour serving $\sigma_{\leq t}$
$T^{\mathrm{return}} \longleftarrow$ shortest possible tour back to the origin unloading every
  currently carried request at its respective source or delivering it to its
  destination
$T^{\mathrm{new}} \longleftarrow$ open/closed tour of length $L(t, 0, \sigma_{\leq t})$ (respectively $L^*(t, 0, \sigma_{\leq t})$)
  starting at 0 and serving $\sigma_{\leq t}$
return $T^{\mathrm{return}} \oplus T^{\mathrm{new}}$
```

First, observe the following lemma stating the simple fact that in the preemptive setting inverting a schedule, i.e., following a schedule back in time, allows to unload requests at their pickup-positions if they were picked up on the schedule.

Lemma 3.1. *In the preemptive setting, suppose at time t^{start} the server is located empty at the origin while all requests are located at their respective sources. If the* *server follows a schedule S after t^{start} until time t^{end}, then the inverse schedule S^{-1} of length $|S^{-1}| = |S| = \big(t^{\text{end}} - t^{\text{start}}\big)$ which returns to the origin in the same way that S has moved out, i.e., for all $t \in \big[t^{\text{end}}, t^{\text{end}} + \big(t^{\text{end}} - t^{\text{start}}\big)\big]$*

$$p^{\text{ALG}}(t) = p^{\text{ALG}}\big(t^{\text{end}} - \big(t - t^{\text{end}}\big)\big),$$

can return every request carried at time t^{end} at its respective source.

Proof. Clearly, if S^{-1} is followed after time t^{end}, then the server ends at

$$p^{\text{ALG}}\big(t^{\text{end}} - \big(\big(t^{\text{end}} + \big(t^{\text{end}} - t^{\text{start}}\big)\big) - t^{\text{end}}\big)\big) = p^{\text{ALG}}\big(t^{\text{start}}\big) = 0.$$

Now suppose that at time t^{end} the server is carrying $k \leq c$ requests. Let $t^{\text{start}} + t_1, ..., t^{\text{start}} + t_k$ be the corresponding times when each of the k requests was picked up at its respective source a_i $(i \in \{1, ..., k\})$. At time $t^{\text{end}} - \big((t^{\text{start}} + t_i) - t^{\text{end}}\big)$ the server is at position a_i again and can unload the corresponding request at its source.

□

3.1.1 ABORT for Open Online Dial-a-Ride

Let us now prove that ABORT is 3-competitive for the preemptive open case.

Theorem 3.2. *Algorithm* ABORT *is* 3*-competitive for preemptive open online Dial-a-Ride and the ratio is tight.*

Proof. Let t_n be the last release time from σ. Further, let t^{start} denote the last time the server is located empty at the origin before t_n and let t_i be the first release time after t^{start}. At time t_i the server aborts its current schedule and returns to the origin by a schedule T^{return}. By Lemma 3.1 we know that the inverse tour of how the server moved after t^{start} allows to return to the origin and unload every at t_i carried request at its respective source in time $(t_i - t^{\text{start}})$. Thus, we have $|T^{\text{return}}| \leq (t_i - t^{\text{start}}) \leq t_i \leq \text{OPT}(\sigma)$ where the last inequality holds as OPT must respect release times. This yields for the completion time of ABORT

$$\text{ABORT}(\sigma) = \underbrace{t_i}_{\leq \text{OPT}(\sigma)} + \underbrace{|T^{\text{return}}|}_{\leq \text{OPT}(\sigma)} + \underbrace{L(t_n, 0, \sigma_{\leq t_n})}_{\leq \text{OPT}(\sigma)} \leq 3\text{OPT}(\sigma).$$

The following TSP-instance shows that this ratio is tight even on the halfline $\mathbb{R}_0^+$ (cf. Figure 3.1). Consider $\sigma = \{(1; 0), (1; 1 - \epsilon)\}$ with some small $\epsilon > 0$.

Right before ABORT is able to serve $(1; 0)$ at time $1 - \epsilon$ the server aborts its schedule to return to the origin only to get back to position 1. This yields a total completion time of $\text{ABORT}(\sigma) = 3 - 2\epsilon$ while we have $\text{OPT}(\sigma) = 1$. Thus, the ratio $\text{ABORT}(\sigma)/\text{OPT}(\sigma)$ can be made arbitrarily close to 3 by choosing ϵ sufficiently small. □

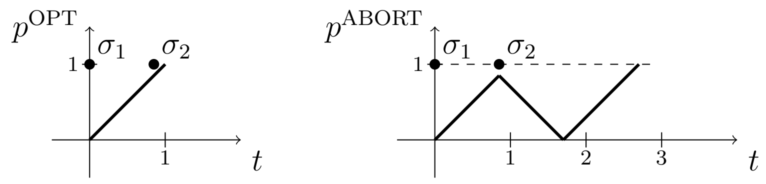

Figure 3.1: TSP-instance $\sigma_1 = (1; 0)$, $\sigma_2 = (1; 1-\epsilon)$ for the open case on the halfline where ABORT is (close to) 3-competitive.

The same competitive ratio holds in the uncapacitated open case.

Theorem 3.3. *Algorithm* ABORT *is* 3*-competitive for uncapacitated open online Dial-a-Ride if one omits unloading requests and the ratio is tight.*

Proof. The proof is identical to the proof of Theorem 3.2 if one omits unloading requests. □

3.1.2 ABORT for Closed Online Dial-a-Ride

Now we prove that ABORT is 2.5-competitive for the preemptive closed case.

Theorem 3.4. *Algorithm* ABORT *is* 2.5*-competitive for preemptive closed online Dial-a-Ride and the ratio is tight.*

Proof. Again, let t_n be the last release time from σ, let t^{start} denote the last time the server is located empty at the origin before t_n and let t_i be the first release time after t^{start}. At time t_i the server aborts its current schedule and returns to the origin by a schedule T^{return}. We distinguish two cases now. If $(t_i - t^{\text{start}}) \leq 1/2\,\text{OPT}(\sigma_{\leq t^{\text{start}}})$, then at t_i going back to the origin in the same way as the server has moved out at t^{start} allows to unload all currently carried requests at their respective sources in time $(t_i - t^{\text{start}})$ by Lemma 3.1. Thus, we have

$$|T^{\text{return}}| \leq 1/2\,\text{OPT}(\sigma). \tag{3.1}$$

If $(t_i - t^{\text{start}}) > 1/2\,\text{OPT}(\sigma_{\leq t^{\text{start}}})$, then we have $|T^{\text{return}}| < 1/2\,\text{OPT}(\sigma_{\leq t^{\text{start}}})$ since completing the tour for $\sigma_{\leq t^{\text{start}}}$ obviously delivers every request carried at time t_i to its destination. Thus, inequality (3.1) holds in both cases and we obtain

$$\text{ABORT}(\sigma) = \underbrace{t_i}_{\leq \text{OPT}(\sigma)} + \underbrace{|T^{\text{return}}|}_{\leq 1/2\,\text{OPT}(\sigma)} + \underbrace{L^*(t_n, 0, \sigma_{\leq t_n})}_{\leq \text{OPT}(\sigma)} \leq 2.5\text{OPT}(\sigma).$$

The following TSP-instance shows that this ratio is tight even on the halfline $\mathbb{R}_0^+$ (cf. Figure 3.2). Consider $\sigma = \{(1;1),(0;2-\epsilon)\}$ with some small $\epsilon > 0$. Right before ABORT is able to serve $(1;1)$ at time $2-\epsilon$ the server aborts its schedule to only return to the origin and then move up to 1 and return to 0 again. This yields a total completion time of $\text{ABORT}(\sigma) = 5 - 2\epsilon$ while we have $\text{OPT}(\sigma) = 2$. Thus, the ratio $\text{ABORT}(\sigma)/\text{OPT}(\sigma)$ can be made arbitrarily close to 2.5 by choosing ϵ sufficiently small. □

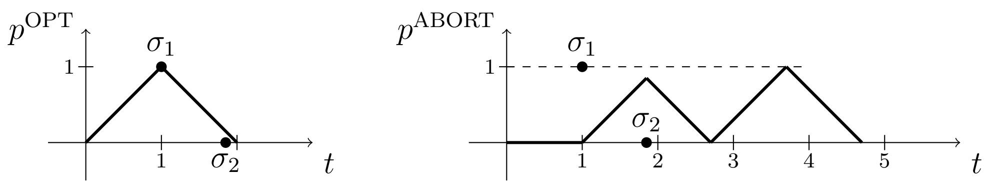

Figure 3.2: TSP-instance $\sigma_1 = (1;1)$, $\sigma_2 = (0;2-\epsilon)$ for the closed case on the halfline where ABORT is (close to) 2.5-competitive.

As in the open version, the same competitive ratio carries over from the preemptive to the uncapacitated case.

Theorem 3.5. *Algorithm* ABORT *is* 2.5*-competitive for uncapacitated closed online Dial-a-Ride if one omits unloading requests and the ratio is tight.*

Proof. The proof is identical to the proof of Theorem 3.4 if one omits unloading requests. □

3.2 Algorithm: ABORT-AND-WAIT

In this section we consider the algorithm ABORT-AND-WAIT (AAW, Algorithm 2) which extends ABORT by a waiting routine: Whenever new requests are released the server also returns to the origin in a shortest possible way while unloading every currently carried request at its respective source. But once at the origin the server waits as long as it can still be θ-competitive for the remaining requests and then starts a shortest schedule for them. Here $\theta \geq 1$ is a scalable waiting parameter. We find that this waiting routine improves the competitiveness of ABORT. In the preemptive (and uncapacitated) open case AAW is $\big(1+\sqrt{2}\big)$-competitive and in the corresponding closed case AAW is 2-competitive for general metric spaces if one chooses suitable $\theta \geq 1$.

We do not claim originality for the algorithm. A similar algorithm has already been investigated by Bjelde *et al.* [1, algorithm 3] for the preemptive open case on the real line. In their version of the algorithm the server instantly unloads all carried requests at its current position once new requests are released and then takes a shortest possible tour back to the origin. This complicates the analysis for general metric spaces since requests can be unloaded at positions which extend the length of an optimal tour starting from 0 in some spaces. In this sense AAW naturally extends their algorithm to be suitable for arbitrary metric spaces in the open case. It also tightens the competitive result of the strategy for the closed case (matching the lower bound) which is not considered separately by Bjelde *et al.* since a best possible algorithm was already known in the closed case (cf. Table 2.1).

Algorithm 2: AAW for Open/Closed online Dial-a-Ride

```
this function is called upon receiving a new request
input  : unserved requests $\sigma_{\leq t}$, current server-position $p^{\text{AAW}}(t)$
output: open/closed tour serving $\sigma_{\leq t}$
$T^{\text{return}} \longleftarrow$ shortest-possible tour back to the origin unloading every
  currently carried request at its respective source
$T^{\text{new}} \longleftarrow$ open/closed tour of length $L(t,0,\sigma_{\leq t})$ (respectively $L^*(t,0,\sigma_{\leq t})$)
  starting at 0 and serving $\sigma_{\leq t}$
return $T^{\text{return}} \oplus \text{waituntil}((\theta-1)\text{OPT}(\sigma_{\leq t})) \oplus T^{\text{new}}$
```

3.2.1 AAW for Open Online Dial-a-Ride

Let us first prove that AAW is $\big(1+\sqrt{2}\big)$-competitive in the preemptive open setting.

Theorem 3.6. *For $\theta = 1+\sqrt{2}$ algorithm* AAW *is $\big(1+\sqrt{2}\big)$-competitive for preemptive open online Dial-a-Ride.*

Proof. Assume that the AAW-server is always able to return to the origin unloading all currently carried requests at their respective sources before time $\sqrt{2}\text{OPT}(\sigma_{\leq t})$ whenever new requests arrive at time t. Once the last request is released at time t_n, the server returns to the origin and waits there until $\sqrt{2}\mathrm{OPT}(\sigma_{\le t_n})$ before starting a new schedule for the remaining requests of length $L(t_n, 0, \sigma_{\le t_n})$. Thus, we obtain

$$\mathrm{AAW}(\sigma) = \sqrt{2}\mathrm{OPT}(\sigma_{\le t_n}) + L(t_n, 0, \sigma_{\le t_n}) \le \big(1+\sqrt{2}\big)\mathrm{OPT}(\sigma).$$

Hence, it suffices to show that the server is always able to return to the origin and unload all currently carried requests at their sources in time $\sqrt{2}\mathrm{OPT}(\sigma_{\le t})$ whenever requests are released at time t. We prove this by induction over the number n of requests. Obviously, the statement holds for $n = 1$ or if the server is already located at the origin at time t_n. Thus, we suppose that the statement is true for $n-1$ for some $n \ge 2$ and that the server is not at the origin at time t_n. Now let t^{start} be the last time before t_n when the AAW-server left the origin being empty and let t_i be the first release time after t^{start}. Note that we have $t^{\text{start}} = \sqrt{2}\mathrm{OPT}(\sigma_{\le t^{\text{start}}})$ by the induction hypothesis. We now distinguish two cases depending on the value of t_i.

Firstly, if $t_i \ge t^{\text{start}} + \mathrm{OPT}(\sigma_{\le t^{\text{start}}}) = \big(1+\sqrt{2}\big)\mathrm{OPT}(\sigma_{\le t^{\text{start}}})$ or equivalently

$$\big(\sqrt{2}-1\big)t_i \ge \mathrm{OPT}(\sigma_{\le t^{\text{start}}}) \tag{3.2}$$

since $1+\sqrt{2} = 1/\big(\sqrt{2}-1\big)$, then at time t_i the AAW-server has just served all requests from $\sigma_{\le t^{\text{start}}}$ and need not unload any requests. Thus, the server returns from a position not further away from the origin than $\mathrm{OPT}(\sigma_{\le t^{\text{start}}})$ in time at most

$$t_i + \mathrm{OPT}(\sigma_{\le t^{\text{start}}}) \le \sqrt{2}(t_i) \le \sqrt{2}\mathrm{OPT}(\sigma_{\le t_n})$$

where the first inequality holds due to (3.2).

Secondly, if $t_i < t^{\text{start}} + \mathrm{OPT}(\sigma_{\le t^{\text{start}}}) = \big(1+\sqrt{2}\big)\mathrm{OPT}(\sigma_{\le t^{\text{start}}})$ or equivalently

$$\big(\sqrt{2}-1\big)t_i < \mathrm{OPT}(\sigma_{\le t^{\text{start}}}), \tag{3.3}$$

then the AAW-server might still carry requests at time t_i. Taking the inverse route back to the origin as the one started at time t^{start} (cf. Lemma 3.1) allows to return all carried requests at their respective sources and arrive at 0 in time

$$t_i + (t_i - t^{\text{start}}) = 2t_i - \sqrt{2}\mathrm{OPT}(\sigma_{\le t^{\text{start}}}) < \sqrt{2}(t_i) \le \sqrt{2}\mathrm{OPT}(\sigma_{\le t_n})$$

where the first inequality is true by (3.3). □

The same competitive ratio holds for the uncapacitated open case.

Theorem 3.7. *For $\theta = 1 + \sqrt{2}$ algorithm* AAW *is $\big(1 + \sqrt{2}\big)$-competitive for uncapacitated open online Dial-a-Ride if one omits unloading requests.*

Proof. The proof is identical to the proof of Theorem 3.6 if one omits unloading requests. $\square$

One might be tempted to greedily force a better competitive ratio by reducing the value of the waiting parameter θ. The following remark states that this is not possible even on the halfline $\mathbb{R}_0^+$.

Remark 3.8. Choosing $\theta \in \big[1, 1 + \sqrt{2}\big)$ yields no better competitive ratio than $1+\sqrt{2}$ for AAW in the open case. Consider the following TSP-instance on $\mathbb{R}_0^+$ (cf. Figure 3.3): First, a request $\sigma_1 = (1; 1)$ is presented. For $\theta \leq 2$, we present as a second request $\sigma_2 = (2; 2)$. In this case, we have $\mathrm{AAW}(\{\sigma_1, \sigma_2\}) = 5$ and $\mathrm{OPT}(\{\sigma_1, \sigma_2\}) = 2$ such that their ratio is 2.5. For $\theta > 2$, we present as a second request $\sigma_2' = (\theta; \theta)$. In this case, we have $\mathrm{AAW}(\{\sigma_1, \sigma_2'\}) = 2\theta + 1$ and $\mathrm{OPT}(\{\sigma_1, \sigma_2'\}) = \theta$. As $\frac{2\theta+1}{\theta}$ is a monotonically decreasing expression for $\theta < 1 + \sqrt{2}$ we obtain a competitive ratio of at least $\frac{2\left(1+\sqrt{2}\right)+1}{1+\sqrt{2}} = 1 + \sqrt{2}$.

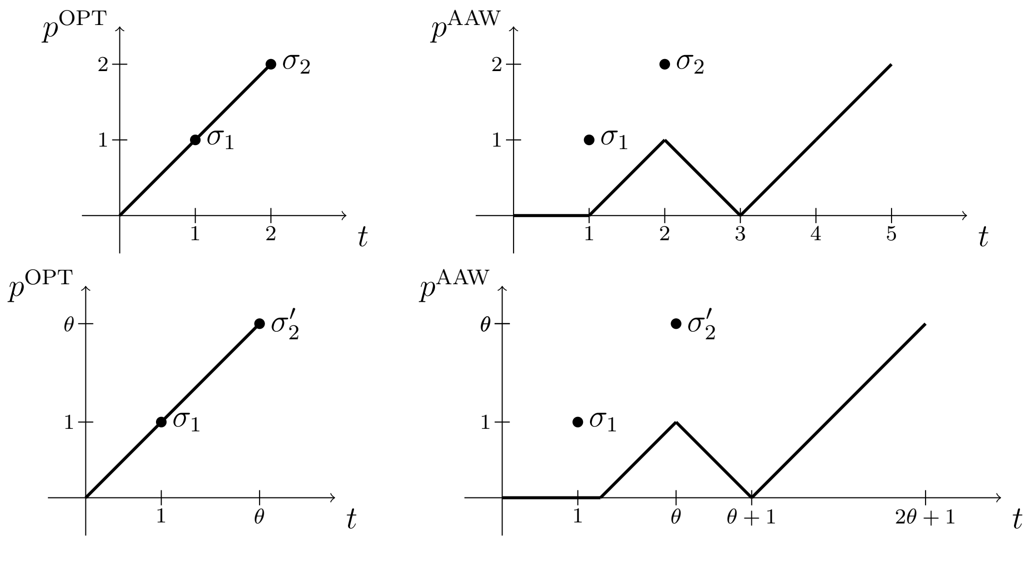

Figure 3.3: At the top: TSP-instance $\sigma_1 = (1; 1)$, $\sigma_2 = (2; 2)$ for the open case on the halfline where AAW with $\theta \leq 2$ is 2.5-competitive.
At the bottom: TSP-instance $\sigma_1 = (1; 1)$, $\sigma_2' = (\theta; \theta)$ for the open case on the halfline where AAW with $2 < \theta < 1 + \sqrt{2}$ is (at least) $\big(1 + \sqrt{2}\big)$-competitive.

3.2.2 AAW for Closed Online Dial-a-Ride

Let us now consider the closed case. We show that AAW is 2-competitive in the preemtive setting.

Theorem 3.9. *For $\theta = 2$ algorithm* AAW *is 2-competitive for preemptive closed online Dial-a-Ride.*

Proof. Let t_n be the last release time of requests from σ. Possibly, after t_n the AAW-server is able to return empty to the origin before time $\mathrm{OPT}(\sigma)$. In this case, the server waits at the origin until time $\mathrm{OPT}(\sigma)$ and then starts a schedule for $\sigma_{\leq t_n}$. This yields

$$\mathrm{AAW}(\sigma) = \mathrm{OPT}(\sigma) + L^*(t_n, 0, \sigma_{\leq t_n}) \leq 2\mathrm{OPT}(\sigma).$$

In contrast to the open case, the server might not be able to return empty to the origin before time $\mathrm{OPT}(\sigma)$. Suppose that we are in this case. In particular, the server is not already located at the origin being empty at time t_n. Now let t^{start} be the last point of time before t_n when the AAW-server left the origin while not carrying any requests. Further, let $t_i \leq t_n$ be the first release time after t^{start}. Note that at time t_i the server starts a shortest possible tour T^{return} to return to the origin and unload all currently carried requests at their respective sources to afterwards serve all remaing requests. We obtain for the completion time of AAW:

$$\mathrm{AAW}(\sigma) = t_i + |T^{\text{return}}| + L^*(t_n, 0, \sigma_{\leq t_n}). \tag{3.4}$$

For the tour returning to the origin, we have

$$|T^{\text{return}}| \leq (t_i - t^{\text{start}}) \tag{3.5}$$

by Lemma 3.1. As a consequence of the triangle inequality and simple properties of $L^*(\cdot,\cdot,\cdot)$, we further obtain

$$\begin{aligned} L^*(t_n, 0, \sigma_{\leq t_n}) &\leq \mathrm{OPT}(\sigma_{\leq t^{\text{start}}}) + L^*(t_i, 0, \sigma_{\geq t_i}) \\ &\leq \mathrm{OPT}(\sigma_{\leq t^{\text{start}}}) + d\big(0, p^{\mathrm{OPT}}(t_i)\big) + L^*\big(t_i, p^{\mathrm{OPT}}(t_i), \sigma_{\geq t_i}\big) \\ &\leq \mathrm{OPT}(\sigma_{\leq t^{\text{start}}}) + 2L^*\big(t_i, p^{\mathrm{OPT}}(t_i), \sigma_{\geq t_i}\big). \end{aligned} \tag{3.6}$$

Plugging (3.5) and (3.6) into (3.4) yields

$$\begin{aligned}\mathrm{AAW}(\sigma) &\leq 2t_i \underbrace{-t^{\mathrm{start}} + \mathrm{OPT}(\sigma_{\leq t^{\mathrm{start}}})}_{\leq 0 \text{ by waiting routine}} + 2L^*\big(t_i, p^{\mathrm{OPT}}(t_i), \sigma_{\geq t_i}\big) \\ &\leq 2t_i + 2L^*\big(t_i, p^{\mathrm{OPT}}(t_i), \sigma_{\geq t_i}\big) \\ &\leq 2\mathrm{OPT}(\sigma)\end{aligned}$$

where the last inequality holds since OPT needs to respect release times. □

Again, the same competitive ratio holds for the uncapacitated closed case.

Theorem 3.10. *For $\theta = 2$ algorithm* AAW *is 2-competitive for uncapacitated closed online Dial-a-Ride if one omits unloading requests.*

Proof. The proof is identical to the proof of Theorem 3.9 if one omits unloading requests. □

Ausiello *et al.* [6, Thm 3.2] have shown that a competitive ratio of 2 is best-possible for preemptive or uncapacitated closed online Dial-a-Ride on general metric spaces. Focusing only on the real line $\mathbb{R}$ or the halfline $\mathbb{R}_0^+$ better competitive ratios than 2 might be possible (cf. Table 2.1). But, as in the open case, greedily reducing the value of the waiting parameter θ does not yield a better competitive ratio in these cases.

Remark 3.11. Choosing $\theta \in [1, 2)$ yields no better competitive ratio than 2 for AAW in the closed case on the halfline $\mathbb{R}_0^+$. Consider the following TSP-instance on $\mathbb{R}_0^+$ (cf. Figure 3.4). First, a request $\sigma_1 = (1; 1)$ is presented. For $\theta \leq 1.5$, we present as a second request $\sigma_2 = (0; 2 - \epsilon)$ for some small $\epsilon > 0$ right before the AAW-server visits position 1. In this case we have $\mathrm{AAW}(\{\sigma_1, \sigma_2\}) = 5 - 2\epsilon$ and $\mathrm{OPT}(\{\sigma_1, \sigma_2\}) = 2$ yielding a competitive ratio arbitrarily close to 2.5 for small enough $\epsilon > 0$. For $\theta > 1.5$, we present as a second request $\sigma_2' = (0; 2\theta - 1 - \epsilon)$. In this case, we have $\mathrm{AAW}(\{\sigma_1, \sigma_2'\}) = 2\theta + 2 - 2\epsilon$ and $\mathrm{OPT}(\{\sigma_1, \sigma_2'\}) = 2\theta - 1 - \epsilon$ yielding a competitive ratio of at least $\frac{6-2\epsilon}{3-\epsilon} = 2$ since $\frac{2\theta+2-2\epsilon}{2\theta-1-\epsilon}$ is a monotonically decreasing expression for $\theta \in [1.5, 2)$ and small enough $\epsilon > 0$.

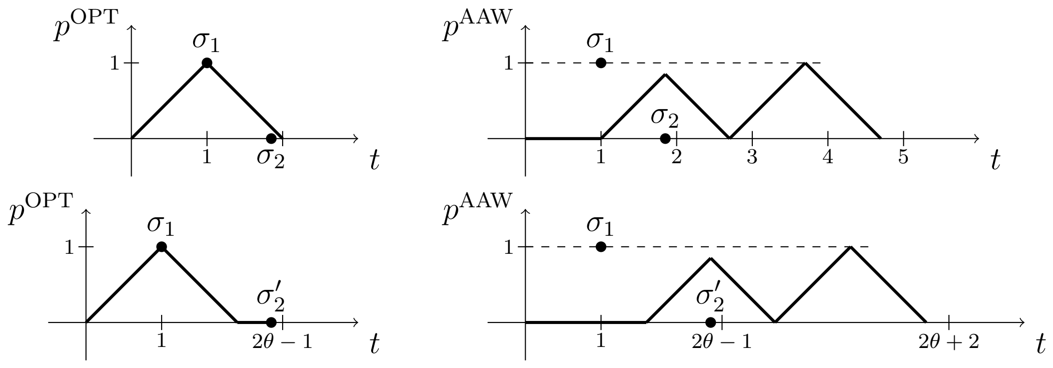

Figure 3.4: At the top: TSP-instance $\sigma_1 = (1;1)$, $\sigma_2 = (2; 2 - \epsilon)$ for the closed case on the halfline where AAW with $\theta \leq 1.5$ is (close to) 2.5-competitive.
At the bottom: TSP-instance $\sigma_1 = (1;1)$, $\sigma_2 = (2\theta - 1 - \epsilon)$ for the closed case on the halfline where AAW with $1.5 < \theta < 2$ is (close to) $\left(1 + \sqrt{2}\right)$-competitive.

3.3 Algorithm: ABORT-OR-REPLAN

In this section we consider the further improved strategy ABORT-OR-REPLAN (AOR): Instead of always aborting a current schedule whenever new requests arrive, the AOR-server only returns to the origin if it can do so in reasonable time. Otherwise, the AOR-server replans its schedule and serves all remaining old and new requests from its current position instead. We only analyze AOR for the uncapacitated closed case on the halfline (see Algorithm 3). In this setting we present a proof that AOR is 1.8536-competitive which improves the former upper bound of 2 established by Ascheuer *et al.* [3, Thm 6] as mentioned earlier. The analysis is not tight in the sense that a better competitive ratio than 1.8536 might be possible by reducing waiting times. The proof is rather technical and heavily depends on the aforementioned setting. AOR might also improve the competitiveness of AAW in other cases but these are not treated in this thesis.

3.3.1 AOR for Uncapacitated Closed Online Dial-a-Ride on the Halfline

Before analyzing the actual algorithm let us first consider the setting of uncapacitated closed online Dial-a-Ride on the halfline: Since the server has infinite capacity, we can demand that whenever it moves over a source of a request, the request is picked up and whenever it is moving over a destination of a request it is currently carrying, the request is delivered. Now on the halfline, if the server is located at the origin, it can serve all released requests in a single closed tour to the uppermost position of the requests. While going up, all requests are picked up and at the latest on the way back down to 0 all requests are eventually delivered to their respective destinations. There is no shorter tour than this for the released requests since the uppermost position must be visited. We denote the uppermost position of all requests released up to time t that are not served if the AOR-server returns to the origin from its current position by

$$p_U(\leq t) := \max\left\{ \max(a_i, b_i) : \begin{array}{l} (a_i, b_i; t_i) \in \sigma_{\leq t} \text{ not currently carried with} \\ b_i \leq p^{\text{AOR}}(t), \text{nor with } b_i \leq a_i \leq p^{\text{AOR}}(t_i) \end{array} \right\}.$$

By $p_U(= t)$ we denote the uppermost position of all requests released at time t, i.e.,

$$p_U(= t) := \max\{\max(a_i, b_i) : (a_i, b_i; t_i) \in \sigma \text{ with } t_i = t\}.$$

Algorithm AOR now works as following: Again, let $\theta \geq 1$ be a scalable waiting parameter. Similar to AAW if at the origin, the AOR-server waits for as long as it can be θ-competitive until time $\theta\text{OPT}(\sigma_{\leq t}) - 2p_U(\leq t)$ before starting a new schedule to the uppermost position. Further, whenever new requests are released and the AOR-server can afford to abort its current schedule and return to the origin while staying θ-competitive for the remaining requests, the server does so. For this purpose we define for the length of the abort-schedule

$$\text{abort}(t) := t + p^{\text{AOR}}(t) + 2p_U(\leq t).$$

If we choose $\theta \in \left[1 + \sqrt{2}/2, 2\right]$, we see that it is not always possible for AOR to abort and stay θ-competitive. In particular, it can happen that while the server is currently on a schedule, started at some time t^{start} at the origin, critical requests are released that do not allow to abort while staying θ-competitive. In such a case, AOR switches to a different strategy depending on the current state of the server: If the AOR-server cannot afford to abort while staying θ-competitive and has not yet been to the uppermost position $p_U(\leq t^{\text{start}})$, then it continues its tour up to $p_U(\leq t^{\text{start}})$ as originally planned. Once there, instead of directly going down back to 0, it serves all requests not served otherwise in a shortest possible way on its tour back to 0. For this purpose let $\sigma^{\text{up}}_{\leq t}$ be the set of requests released before or at time $t \leq t^{\text{start}} + p_U(\leq t^{\text{start}})$ and not served once the AOR-server arrives at $p_U(\leq t^{\text{start}})$, i.e.,

$$\sigma_{\leq t}^{\text{up}} = \left\{ (a_i, b_i; t_i) \in \sigma_{\leq t} : \begin{array}{l} \text{not carried with } p^{\text{AOR}}(t) \leq b_i \leq p_U(\leq t^{\text{start}}), \\ \text{nor } p^{\text{AOR}}(t) \leq a_i \leq b_i \leq p_U(\leq t^{\text{start}}) \end{array} \right\}.$$

Otherwise, if the AOR-server cannot afford to abort while staying θ-competitive and has already been to $p_U(\leq t^{\text{start}})$, then it immediately starts a new shortest tour for the remaining requests setting off from its current position. We find that in both cases AOR finishes serving all requests and returns to the origin in time $(1+\theta/2)\text{OPT}(\sigma)$ if $\theta \geq 1+\sqrt{2}/2 \approx 1.7071$ yielding a competitive ratio of roughly 1.8536.

It might be possible to greedily reduce the value of the waiting parameter $\theta \geq 1$ below $1+\sqrt{2}/2$ to obtain an improved competitive ratio for AOR. However, our proof for the competitive ratio of 1.8536 heavily depends on Lemma 3.14 which only holds for $\theta \geq 1+\sqrt{2}/2$.

Algorithm 3: AOR for Closed Online Dial-a-Ride on the Halfline ($c = \infty$)

```
this function is called upon receiving a new request
input  : unserved requests $\sigma_{\leq t}$, current server-position $p^{\text{AAW}}(t)$, last time
         $t^{\text{start}}$ such that the server left 0 with abort($t^{\text{start}}$) $= \theta\text{OPT}(\sigma_{\leq t^{\text{start}}})$
output: closed tour serving $\sigma_{\leq t}$
if abort($t$) $\leq \theta\text{OPT}(\sigma_{\leq t})$ then
    $t^{\text{start}}_{\text{new}} \longleftarrow \theta\text{OPT}(\sigma_{\leq t}) - 2p_U(\leq t)$
    return move(0) $\oplus$ waituntil($t^{\text{start}}_{\text{new}}$) $\oplus$ move($p_U(\leq t)$) $\oplus$ move(0)
else if $t < t^{\text{start}} + p_U(\leq t^{\text{start}})$ then
    $T^{\text{DOWN}} \longleftarrow$ tour of length $L^*\big(t, p_U(\leq t^{\text{start}}), \sigma^{\text{up}}_{\leq t}\big)$
    return move($p_U(\leq t^{\text{start}})$) $\oplus$ $T^{\text{DOWN}}$
else
    $T^{\text{DOWN}} \longleftarrow$ tour of length $L^*\big(t, p^{\text{AOR}}(t), \sigma_{\leq t}\big)$
    return $T^{\text{DOWN}}$
```

The following theorem shows that for $\theta \in \big[1+\sqrt{2}/2, 2\big]$ AOR is $(1+\theta/2)$-competitive. The proof uses Lemmas and Corollarys 3.13-3.19.

Theorem 3.12. *For* $\theta \in \big[1+\sqrt{2}/2, 2\big]$ *algorithm* AOR *is* $(1+\theta/2)$*-competitive and the ratio is tight.*

Proof. First of all, note that for the first release time t_1 we find abort$(t_1) \leq \theta\text{OPT}(\sigma_{\leq t_1})$ by Lemma 3.14. By Lemma 3.17 we then know that the AOR-server leaves the origin for the first time at a time t' with $\mathrm{abort}(t') = \theta\mathrm{OPT}(\sigma_{\leq t'})$. As we now know that such a time exists, let t^{start} be the last point of time when the AOR-server leaves the origin with $\mathrm{abort}(t^{\mathrm{start}}) = \theta\mathrm{OPT}(\sigma_{\leq t^{\mathrm{start}}})$. For all $t > t^{\mathrm{start}}$ we now have $\mathrm{abort}(t) > \theta\mathrm{OPT}(\sigma_{\leq t})$. Otherwise Corollary 3.16 and Lemma 3.17 would contradict the assumption that t^{start} is the last time the AOR-server leaves the origin with $\mathrm{abort}(t^{\mathrm{start}}) = \theta\mathrm{OPT}(\sigma_{\leq t^{\mathrm{start}}})$.

This allows us to obtain properties for the requests released after t^{start}. Firstly, note that by the contraposition of Lemma 3.18 (ii) we know that no request is released after $t^{\mathrm{start}} + 2p_U(\leq t^{\mathrm{start}})$. Further, not all requests released after t^{start} have an influence on the concrete movement of the AOR-server. Consider e.g. the following two cases:

- For requests $(a_i, b_i; t_i) \in \sigma_{>t^{\mathrm{start}}}$ with $t_i \in (t^{\mathrm{start}}, t^{\mathrm{start}} + p_U(\leq t^{\mathrm{start}})]$ and $p^{\mathrm{AOR}}(t_i) \leq a_i < b_i$ we have $a_i < b_i < p_U(\leq t^{\mathrm{start}})$ by the contraposition of Lemma 3.14. Thus, all such requests are served while the AOR-server moves towards $p_U(\leq t^{\mathrm{start}})$.

- For requests $(a_i, b_i; t_i) \in \sigma_{>t^{\mathrm{start}}}$ with $b_i \leq a_i$ we distinguish two possibilities. If such a request is released while the server is moving upwards to $p_U(\leq t^{\mathrm{start}})$, i.e., $t_i \in (t^{\mathrm{start}}, t^{\mathrm{start}} + p_U(\leq t^{\mathrm{start}})]$, then any tour back from $p_U(\leq t^{\mathrm{start}})$ to the origin serves it since $b_i \leq a_i < p_U(\leq t^{\mathrm{start}}) = p^{\mathrm{AOR}}(t^{\mathrm{start}} + p_U(\leq t^{\mathrm{start}}))$ by the contraposition of Lemma 3.14. Otherwise, if such a request is released after the server has been to position $p_U(\leq t^{\mathrm{start}})$, i.e., $t_i \in (t^{\mathrm{start}} + p_U(\leq t^{\mathrm{start}}), t^{\mathrm{start}} + 2p_U(\leq t^{\mathrm{start}})]$, then by the contraposition of Lemma 3.18 (iii) we have $b_i \leq a_i \leq p^{\mathrm{AOR}}(t_i)$ and the request is also served by any route back to the origin.

These considerations motivate the following definition of the set of relevant requests σ^{up} that are released while the server is moving upwards to $p_U(\leq t^{\mathrm{start}})$, respectively σ^{down} for the relevant requests released after the server visited $p_U(\leq t^{\mathrm{start}})$ at time $t^{\mathrm{start}} + p_U(\leq t^{\mathrm{start}}) =: t^{\mathrm{top}}$:

$$\sigma^{\mathrm{up}} := \left\{(a_i, b_i; t_i) \in \sigma : t_i \in \left(t^{\mathrm{start}}, t^{\mathrm{top}}\right], a_i < p^{\mathrm{AOR}}(t_i), b_i\right\},$$

$$\sigma^{\mathrm{down}} := \left\{(a_i, b_i; t_i) \in \sigma : t_i \in \left(t^{\mathrm{top}}, t^{\mathrm{start}} + 2p_U(\leq t^{\mathrm{start}})\right], a_i < b_i\right\}.$$

Taking the considerations from above we conclude: If the AOR-server serves all requests from $\sigma^{\text{up}} \cup \sigma^{\text{down}}$ during its schedule down to 0 from $p_U(\leq t^{\text{start}})$, it already serves all requests from $\sigma_{>t^{\text{start}}}$ in the process.

Let us now investigate how much time AOR needs to serve all requests from $\sigma^{\text{up}} \cup \sigma^{\text{down}}$. Once the AOR-server arrives at $p_U(\leq t^{\text{start}})$ at time $t^{\text{top}} = t^{\text{start}} + p_U(\leq t^{\text{start}})$, it follows a shortest schedule back to the origin serving unserved requests of length $L^*(t^{\text{top}}, p_U(\leq t^{\text{start}}), \sigma_{\leq t^{\text{top}}}) = L^*(t^{\text{top}}, p_U(\leq t^{\text{start}}), \sigma^{\text{up}})$. Because every request $(a_i, b_i; t_i) \in \sigma^{\text{down}}$ satisfies $a_i < b_i \leq p_U(\leq t^{\text{start}}) - (t_i - t^{\text{top}})$ by the contraposition of Lemma 3.18 (iii), we obtain from Lemma 3.19 that the unique optimal tours for $\sigma^{\text{up}} \cup \{(a_j, b_j; t_j) \in \sigma^{\text{down}} : t_j < t_i\}$ and $\sigma^{\text{up}} \cup \{(a_j, b_j; t_j) \in \sigma^{\text{down}} : t_j < t_i\} \cup \{(a_i, b_i; t_i)\}$ coincide until time t_i. Iterating this argument for each request from σ^{down} yields that whenever a new request from σ^{down} is released, AOR can adapt its route to still be optimal in the sense that it completes its tour for $\sigma^{\text{up}} \cup \sigma^{\text{down}}$ in time $t^{\text{top}} + L^*(t^{\text{top}}, p_U(\leq t^{\text{start}}), \sigma^{\text{up}} \cup \sigma^{\text{down}})$. Thus, we find for the completion time of AOR:

$$\text{AOR}(\sigma) = t^{\text{start}} + p_U(\leq t^{\text{start}}) + L^*(t^{\text{top}}, p_U(\leq t^{\text{start}}), \sigma^{\text{up}} \cup \sigma^{\text{down}}). \tag{3.7}$$

For better readability, we now write L^* for $L^*(t^{\text{top}}, p_U(\leq t^{\text{start}}), \sigma^{\text{up}} \cup \sigma^{\text{down}})$ and p_U for $p_U(\leq t^{\text{start}})$.

Let us now take a look at the completion time of OPT. We can assume that OPT serves no request from σ^{up} or σ^{down} before visiting p_U: If OPT does serve a request $(a_j, b_j; t_j) \in \sigma^{\text{up}} \cup \sigma^{\text{down}}$ before visiting p_U, we get $\text{OPT}(\sigma) \geq t^{\text{start}} + 2p_U$ since the OPT-server picks up the request at a_j after the AOR-server had been there and then still needs to go to p_U and back to 0. Hence, together with (3.7) we get

$$\begin{aligned}
\frac{\text{AOR}(\sigma)}{\text{OPT}(\sigma)} &\leq \frac{t^{\text{start}} + 2p_U + (L^* - p_U)}{t^{\text{start}} + 2p_U} && \text{by (3.7)} \\
&\leq 1 + \frac{\overbrace{(L^* - p_U)}^{\leq 2p_U}}{\underbrace{t^{\text{start}} + 2p_U}_{=\text{abort}(t^{\text{start}}) = \theta\text{OPT}(\sigma) \geq 2\theta p_U}} \\
&\leq 1 + \frac{2p_U}{2\theta p_U} \\
&= 1 + \frac{1}{\theta} \leq 1 + \theta/2
\end{aligned}$$

where the last inequality holds if $\theta \geq \sqrt{2} \approx 1.41$ which completes the proof. Consequently, we can indeed assume that OPT serves no request from σ^{up} or σ^{down} before visiting p_U. Thus, we have

$$\text{OPT}(\sigma) \geq 2p_U + (L^* - p_U). \tag{3.8}$$

For the next part of the proof let "diff" denote the extra time that OPT needs to serve the incoming requests from after t^{start}, i.e., $\text{diff} := \text{OPT}(\sigma) - \text{OPT}(\sigma_{\leq t^{\text{start}}})$. Note that $(L^* - p_U)$ denotes the extra time it takes AOR to serve the incoming requests from after t^{start}. Possibly OPT needs more extra time to serve the requests from after t^{start} than AOR. We assume that we even have $\theta\text{diff} \geq (L^* - p_U)$. We then obtain

$$\text{AOR}(\sigma) = \underbrace{t^{\text{start}} + 2p_U}_{=\text{abort}(t^{\text{start}})=\theta\text{OPT}(\sigma_{\leq t^{\text{start}}})} + \underbrace{(L^* - p_U)}_{\leq\theta\text{diff}} \leq \theta\text{OPT}(\sigma).$$

Thus, we can assume that we have $\theta\text{diff} < (L^* - p_U)$. Here we need to distinguish two cases depending on the value of $(L^* - p_U)$.

Case 1. Suppose that $(L^* - p_U) \leq \frac{4-2\theta}{\theta}p_U$. In this case OPT needs less extra time to serve the requests released after t^{start} than AOR. This may happen if e.g. OPT was waiting at some point during its schedule for $\sigma_{\leq t^{\text{start}}}$ and can now use this time for good to serve requests from after t^{start}. Let "prepared" denote the difference between the extra time it takes to serve the requests from after t^{start} for AOR and OPT, i.e.,

$$\text{prepared} := (L^* - p_U) - \text{diff}.$$

From (3.8) and

$$(L^* - p_U) = \text{prepared} + \text{diff} = \text{prepared} + \text{OPT}(\sigma) - \text{OPT}(\sigma_{\leq t^{\text{start}}})$$

we obtain

$$\text{OPT}(\sigma_{\leq t^{\text{start}}}) \geq 2p_U + \text{prepared}. \tag{3.9}$$

Using (3.7), (3.9) and the case assumption $(L^* - p_U) \leq \frac{4-2\theta}{\theta}p_U$ we get for the ratio of $\text{AOR}(\sigma)$ and $\text{OPT}(\sigma)$:

$$\begin{aligned}
\frac{\mathrm{AOR}(\sigma)}{\mathrm{OPT}(\sigma)} &= \frac{t^{\text{start}} + 2p_U + (L^* - p_U)}{\mathrm{OPT}(\sigma_{\leq t^{\text{start}}}) + \text{diff}} && \text{by (3.7)} \\
&= \frac{\theta \mathrm{OPT}(\sigma_{\leq t^{\text{start}}}) + (L^* - p_U)}{\mathrm{OPT}(\sigma_{\leq t^{\text{start}}}) + \text{diff}} \\
&\leq \frac{\theta\Big(2p_U + \overbrace{\text{prepared}}^{\leq (L^*-p_U)}\Big) + (L^* - p_U)}{2p_U + \underbrace{\text{prepared} + \text{diff}}_{=(L^*-p_U)}} && \text{mon. incr. expr., (3.9)} \\
&\leq \frac{\theta(2p_U + (L^* - p_U)) + (L^* - p_U)}{2p_U + (L^* - p_U)} \\
&= \theta + \frac{(L^* - p_U)}{2p_U + (L^* - p_U)} \\
&\leq \theta + \frac{\frac{4-2\theta}{\theta}}{\frac{4}{\theta}} && \text{mon. incr. expr., (ass)} \\
&= 1 + \frac{\theta}{2}.
\end{aligned}$$

Note that for the first inequality the expression is monotonically increasing for $\mathrm{OPT}(\sigma_{\leq t^{\text{start}}})$ since we assumed that $\theta\text{diff} < (L^* - p_U)$.

Case 2. Suppose that $(L^* - p_U) > \frac{4-2\theta}{\theta} p_U$. Let t_i be the first release time after t^{start}. First, let us assume that $t_i \leq 2p_U$. From (3.7), (3.8) and the case assumption $(L^* - p_U) > \frac{4-2\theta}{\theta} p_U$ we obtain

$$\begin{aligned}
\frac{\mathrm{AOR}(\sigma)}{\mathrm{OPT}(\sigma)} &\leq \frac{t^{\text{start}} + 2p_U + (L^* - p_U)}{2p_U + (L^* - p_U)} && \text{by (3.7), (3.8)} \\
&= 1 + \frac{\overbrace{t^{\text{start}}}^{\leq t_i \leq 2p_U}}{2p_U + \underbrace{(L^* - p_U)}_{> \frac{4-2\theta}{\theta} p_U}} \\
&\leq 1 + \frac{2p_U}{2p_U + \frac{4-2\theta}{\theta} p_U} = 1 + \frac{\theta}{2}.
\end{aligned}$$

Now let us assume that $t_i > 2p_U$. Here we note

$$\mathrm{OPT}(\sigma) \geq t_i + (L^* - p_U) \tag{3.10}$$

as OPT needs to respect release times and $(L^* - p_U)$ denotes the length of the intervals that every tour for $\sigma^{\text{up}} \cup \sigma^{\text{down}}$ needs to traverse (cf. Lemma 3.19). Using

(3.7), (3.10) and the case assumption $(L^* - p_U) > \frac{4-2\theta}{\theta} p_U$ we get

$$\begin{aligned}
\frac{\mathrm{AOR}(\sigma)}{\mathrm{OPT}(\sigma)} &\leq \frac{t^{\mathrm{start}} + 2p_U + (L^* - p_U)}{t_i + (L^* - p_U)} && \text{by (3.7), (3.10)} \\
&\leq \frac{t_i + 2p_U + (L^* - p_U)}{t_i + (L^* - p_U)} \\
&\leq \frac{4p_U + \frac{4-2\theta}{\theta} p_U}{2p_U + \frac{4-2\theta}{\theta} p_U} && \text{mon. decr. expr., (ass)} \\
&= 1 + \frac{\theta}{2}.
\end{aligned}$$

In order to see that the ratio of $(1 + \theta/2)$ is tight for the non-trivial cases $\theta < 2$, consider the following Dial-a-Ride instance for $\theta < 2$ (cf. Figure 3.5). At time 1 two requests are released: $\sigma_1 = (1, 1; 1)$ and $\sigma_2 = \left(0, \frac{4-2\theta}{2\theta}; 1\right)$. Note that $\mathrm{OPT}(\{\sigma_1, \sigma_2\}) = 2 + \frac{4-2\theta}{\theta}$ as the OPT-server can already be at position 1 at time 1. Since $\theta \cdot \left(2 + \frac{4-2\theta}{\theta}\right) = 4$, the AOR-server starts its tour towards position 1 at time 2. Right after the AOR-server leaves the origin, at time $2 + \epsilon$ for some small $\epsilon > 0$, a new request is presented $\sigma_3 = \left(0, \frac{4-2\theta}{2\theta}; 2 + \epsilon\right)$. We obtain $\mathrm{OPT}(\{\sigma_1, \sigma_2, \sigma_3\}) = 2 + \epsilon + \frac{4-2\theta}{\theta}$. AOR does not abort its schedule towards position 1 at time $2 + \epsilon$ since

$$\frac{\mathrm{abort}(2+\epsilon)}{\mathrm{OPT}(\{\sigma_1, \sigma_2, \sigma_3\})} = \frac{2 + 2\epsilon + 2}{2 + \epsilon + \frac{4-2\theta}{\theta}} = \frac{\theta\left(2 + \frac{4-2\theta}{\theta}\right) + 2\epsilon}{2 + \frac{4-2\theta}{\theta} + \epsilon} > \theta$$

for $\theta < 2$. Thus, we get $\mathrm{AOR}(\{\sigma_1, \sigma_2, \sigma_3\}) = 4 + \frac{4-2\theta}{\theta}$ yielding a competitive ratio of at least $\left(4 + \frac{4-2\theta}{\theta}\right) / \left(2 + \epsilon + \frac{4-2\theta}{\theta}\right)$ which can be made arbitrarily close to $(4 + \frac{4-2\theta}{\theta})/(2 + \frac{4-2\theta}{\theta}) = 1 + \theta/2$ by choosing $\epsilon > 0$ sufficiently small. $\square$

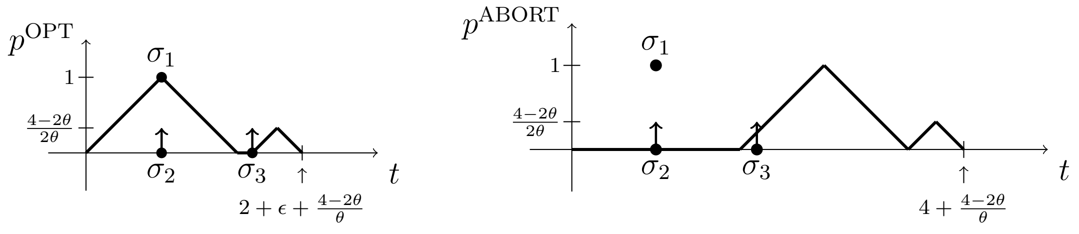
p^{OPT}
σ_1
1
$\frac{4-2\theta}{2\theta}$
σ_2
σ_3
t
$2 + \epsilon + \frac{4-2\theta}{\theta}$
p^{ABORT}
σ_1
1
$\frac{4-2\theta}{2\theta}$
σ_2
σ_3
t
$4 + \frac{4-2\theta}{\theta}$

Figure 3.5: DaR-instance $\sigma_1 = (1, 1; 1)$, $\sigma_2 = \left(0, \frac{4-2\theta}{2\theta}; 1\right)$, $\sigma_3 = \left(0, \frac{4-2\theta}{2\theta}; 2 + \epsilon\right)$ for the closed case on the halfline where AOR with $\theta < 2$ is (close to) $(1 + \theta/2)$-competitive.

The following lemma states the simple fact that if the AOR-server is aborting a schedule while new requests arrive that do not force AOR to move any differently than it planned anyway, then AOR can keep aborting and still be θ-competitive for the remaining requests.

Lemma 3.13. *If at time t_j requests are released with $p_U(= t_j) \leq p_U(\leq t_{j-1})$, as well as* $\text{abort}(t_{j-1}) \leq \theta\text{OPT}(\sigma_{\leq t_{j-1}})$ *and if the AOR-server is moving downwards since t_{j-1}, then* $\text{abort}(t_j) \leq \theta\text{OPT}(\sigma_{\leq t_j})$.

Proof. As $p_U(= t_j) \leq p_U(\leq t_{j-1})$, the planned abort-tour towards $p_U(\leq t_{j-1})$ suffices to also serve all requests released at t_j. Further, as the AOR-server is moving downwards since time t_{j-1} we have

$$p^{\text{AOR}}(t_j) = p^{\text{AOR}}(t_{j-1}) - (t_j - t_{j-1}). \tag{3.11}$$

From this we obtain

$$\begin{aligned} \text{abort}(t_j) &= t_j + p^{\text{AOR}}(t_j) + 2p_U(\leq t_{j-1}) \\ &= t_{j-1} + p^{\text{AOR}}(t_{j-1}) + 2p_U(\leq t_{j-1}) && \text{by (3.11)} \\ &= \text{abort}(t_{j-1}) \\ &\leq \theta\text{OPT}(\sigma_{\leq t_{j-1}}) \leq \theta\text{OPT}(\sigma_{\leq t_j}). \end{aligned}$$

□

Whenever a new extreme request is released and θ is chosen sufficiently large, the following lemma shows that AOR can abort its schedule and be θ-competitive for the remaining requests.

Lemma 3.14. *If $\theta \geq 1 + \sqrt{2}/2$ and at time t_j a new uppermost request is released, i.e., $p_U(= t_j) \geq p_U(\leq t_{j-1})$, then the* AOR*-server can* abort *its current schedule and still be θ-competitive for $\sigma_{\leq t_j}$, i.e.,* $\text{abort}(t_j) \leq \theta\text{OPT}(\sigma_{\leq t_j})$.

Proof. First, we consider the case that the AOR-server is located at the origin at time t_j. Then, we have $\text{abort}(t_j) = t_j + 2p_U(= t_j)$ while
$\text{OPT}(\sigma_{\leq t_j}) \geq \max(t_j + p_U(= t_j), 2p_U(= t_j))$. For $t_j \leq p_U(= t_j)$ we get

$$\frac{\text{abort}(t_j)}{\text{OPT}(\sigma_{\leq t_j})} \leq \frac{3p_U(= t_j)}{2p_U(= t_j)} = 1.5.$$

And similarly for $t_j > p_U(= t_j)$ we have

$$\frac{\text{abort}(t_j)}{\text{OPT}(\sigma_{\leq t_j})} \leq \frac{t_j + 2p_U(= t_j)}{t_j + p_U(= t_j)} \leq \frac{3p_U(= t_j)}{2p_U(= t_j)} = 1.5$$

where the last inequality holds since the expression is monotonically decreasing for $t_j \geq p_U(= t_j)$.

Hence, we can assume that AOR is not at the origin at time t_j. Let t^{start} be the last point of time before t_j when the AOR-server left the origin to visit $p_U(\leq t^{\text{start}})$. Also let $\lambda \geq 1$ such that $\lambda p_U(\leq t^{\text{start}}) = p_U(= t_j)$. For the proof we need to assume that we have

$$p^{\text{AOR}}(t_j) \leq p_U(\leq t^{\text{start}}). \tag{3.12}$$

This is obviously the case if t_j is the first release time after t^{start} with requests above the current extreme $p_U(\leq t^{\text{start}})$. If t_j is not the first release time after t^{start} with new extreme requests, an inductive argument on why (3.12) holds nevertheless is given at the end.

For the moment we assume that we have (3.12). We now distinguish two cases depending on the value of λ.

Case 1. For $\lambda \leq 1 + \sqrt{2}$ we first of all note that since the server would have waited otherwise, we have

$$t^{\text{start}} \geq \theta \text{OPT}(\sigma_{\leq t^{\text{start}}}) - 2p_U(\leq t^{\text{start}}) \geq (2\theta - 2)p_U(\leq t^{\text{start}}).$$

For the current time $t_j \geq t^{\text{start}} + p^{\text{AOR}}(t_j)$ this implies

$$t_j \geq (2\theta - 2)p_U(\leq t^{\text{start}}) + p^{\text{AOR}}(t_j). \tag{3.13}$$

By (3.12), (3.13) and the case-assumption $\lambda \leq 1 + \sqrt{2}$ we obtain for the ratio of abort(t_j) and OPT$(\sigma_{\leq t_j})$

$$\begin{aligned}
\frac{\text{abort}(t_j)}{\text{OPT}(\sigma_{t_j})} &\leq \frac{t_j + p^{\text{AOR}}(t_j) + 2\lambda p_U(\leq t^{\text{start}})}{t_j + \lambda p_U(\leq t^{\text{start}})} \\
&\leq \frac{(2\theta + 2\lambda - 2)p_U(\leq t^{\text{start}}) + 2p^{\text{AOR}}(t_j)}{(2\theta + \lambda - 2)p_U(\leq t^{\text{start}}) + p^{\text{AOR}}(t_j)} && \text{mon. decr. expr., (3.13)} \\
&\leq \frac{2\theta + 2\lambda}{2\theta + \lambda - 1} && \text{mon. incr. expr., (3.12)} \\
&\leq \frac{2\theta + 2(1 + \sqrt{2})}{2\theta + \sqrt{2}} && \text{mon. incr. expr., (ass)} \\
&\leq \theta
\end{aligned}$$

where the last inequality holds only for $\theta \geq 1 + \sqrt{2}/2$.

Case 2. For $\lambda > 1 + \sqrt{2}$ we distinguish the following two cases depending on the value of t_j. If we have

$$t_j \geq \lambda p_U(\leq t^{\text{start}}), \tag{3.14}$$

then we obtain by (3.12), (3.14) and the case-assumption $\lambda > 1 + \sqrt{2}$ that

$$\begin{aligned}
\frac{\text{abort}(t_j)}{\text{OPT}(\sigma_{t_j})} &\leq \frac{t_j + p^{\text{AOR}}(t_j) + 2\lambda p_U(\leq t^{\text{start}})}{t_j + \lambda p_U(\leq t^{\text{start}})} \\
&\leq \frac{3\lambda p_U(\leq t^{\text{start}}) + p^{\text{AOR}}(t_j)}{2\lambda p_U(\leq t^{\text{start}})} && \text{mon. decr. expr., (3.14)} \\
&\leq \frac{3\lambda + 1}{2\lambda} && \text{by (3.12)} \\
&\leq \frac{3\big(1+\sqrt{2}\big)+1}{2\big(1+\sqrt{2}\big)} && \text{mon. decr. expr., (ass)} \\
&= 1 + \frac{\sqrt{2}}{2} \leq \theta.
\end{aligned}$$

Similarly if we have $t_j < \lambda p_U(\leq t^{\text{start}})$, we get

$$\begin{aligned}
\frac{\text{abort}(t_j)}{\text{OPT}(\sigma_{t_j})} &\leq \frac{t_j + p^{\text{AOR}}(t_j) + 2\lambda p_U(\leq t^{\text{start}})}{2\lambda p_U(\leq t^{\text{start}})} \\
&< \frac{3\lambda p_U(\leq t^{\text{start}}) + p^{\text{AOR}}(t_j)}{2\lambda p_U(\leq t^{\text{start}})} \\
&\leq \theta
\end{aligned}$$

by the same reasoning as above.

To complete the proof let us now show that if t_j is the k-th release time after t^{start} with new extreme requests and $k \geq 2$, then (3.12) still holds at this point of time. For this purpose suppose that (3.12) is true for the $(k-1)$-th release time after t^{start} with new extreme requests. Let this release time be given by t_{j-h}. By the considerations from above we know that $\text{abort}(t_{j-h}) \leq \theta\text{OPT}(\sigma_{\leq t_{j-h}})$. Inductively for every release time after t_{j-h} the server keeps aborting its schedule and continues to move down either by Lemma 3.13 if the requests are below the current extreme or by the above considerations since (3.12) was already true at time t_{j-h} and the server only moved down since. □

Note that Lemma 3.14 in fact holds only if $\theta \geq 1 + \sqrt{2}/2$ as stated by the following remark.

Remark 3.15. If we have $\theta \in \big[1, 1+\sqrt{2}/2\big)$, there exists a request-sequence such that the AOR-server cannot abort its current schedule and stay θ-competitive for the remaining requests. Consider the following TSP-instance (cf. Figure 3.6). As a first request $\sigma_1 = (1;0)$ is released. When this request is served by AOR at time $2\theta - 1$, a new request $\sigma_2 = (2\theta-1; 2\theta-1)$ is released. At this time the length of the abort-schedule for AOR is given by $\text{abort}(2\theta-1) = 6\theta - 2$ while we have $\text{OPT}(\{\sigma_1,\sigma_2\}) = 4\theta - 2$. The least value for $\theta \geq 1$ that satisfies $\frac{6\theta-2}{4\theta-2} \leq \theta$ is $1+\sqrt{2}/2$.

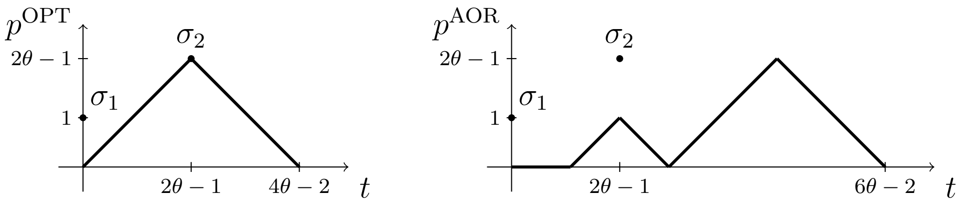

Figure 3.6: TSP-instance $\sigma_1 = (1;0)$, $\sigma_2 = (2\theta-1; 2\theta-1)$ for the closed uncapacitated case on the halfline where we suppose that the AOR-server aborts its schedule at time $2\theta - 1$.

Lemma 3.14 allows us to extend Lemma 3.13 such that we do not need to assume that new requests are released only below the current uppermost position. Once the AOR-server aborts a schedule, it does not change its decision. It returns to the origin being θ-competitive whatever requests are released. Formally, we obtain the following corollary.

Corollary 3.16. *If $\theta \geq 1+\sqrt{2}/2$ and if* $\text{abort}(t) \leq \theta\text{OPT}(\sigma_{\leq t})$ *for some $t \geq 0$, then the* AOR*-server arrives at the origin at time $t' = t + p^{\text{AOR}}(t)$ with* $\text{abort}(t') \leq \theta\text{OPT}(\sigma_{\leq t'})$.

The next lemma assures that while the AOR-server is located at the origin and the currently planned schedule for the remaining requests is θ-competitive, no request can make AOR lose its θ-competitiveness if θ is sufficiently large.

Lemma 3.17. *If $\theta \geq 3/2$ and if the* AOR*-server is currently located at the origin at time t with* $\text{abort}(t) \leq \theta\text{OPT}(\sigma_{\leq t})$*, then* AOR *leaves the origin at a time t^{start} for which* $\text{abort}(t^{\text{start}}) = \theta\text{OPT}(\sigma_{\leq t^{\text{start}}})$.

Proof. Note that if no requests arrive while the AOR-server is at the origin, then AOR waits until time $\theta\text{OPT}(\sigma_{\leq t}) - 2p_U(\leq t) =: t^{\text{start}}$ at which it leaves the origin and the claim obviously holds.

Thus, suppose the server is waiting up to some time $t_j > t$ when new requests are released. Recall from the first part of the proof of Lemma 3.14 that $p_U(= t_j) \geq p_U(\leq t)$ implies $\text{abort}(t_j) \leq \frac{3}{2}\text{OPT}(\sigma_{\leq t_j})$. Hence, assume that $p_U(= t_j) < p_U(\leq t)$. In this case the planned tour towards $p_U(\leq t)$ suffices to serve all requests from time t_j. As the server was waiting at the origin up to time t_j with knowing the requests from $\sigma_{\leq t}$ we have $t_j \leq \theta\text{OPT}(\sigma_{\leq t}) - 2p_U(\leq t)$ which yields

$$\text{abort}(t_j) = t_j + 2p_U(\leq t) \leq \theta\text{OPT}(\sigma_{\leq t}) \leq \theta\text{OPT}(\sigma_{\leq t_j}).$$

Thus, whenever new requests are released at a time t_j we still find $\text{abort}(t_j) \leq \theta\text{OPT}(\sigma_{\leq t_j})$ and the server either continues to wait after t_j if $\text{abort}(t_j) < \theta\text{OPT}(\sigma_{\leq t_j})$ or it leaves the origin if $\text{abort}(t_j) = \theta\text{OPT}(\sigma)$. Iterating this argument until no requests are released while the server is waiting yields that eventually the server leaves the origin at a time t^{start} for which $\text{abort}(t^{\text{start}}) = \theta\text{OPT}(\sigma_{\leq t^{\text{start}}})$. $\square$

The next lemma is used to obtain properties for the requests that are released once the server cannot abort its schedule anymore.

Lemma 3.18. *If $\theta \in \left[1 + \sqrt{2}/2, 2\right]$ and the AOR-server leaves the origin to start a new schedule at time t^{start}, then the following statements hold:*

(i) *If requests are released at a time $t_j > t^{\text{start}}$ with $p_U(= t_j) \geq \frac{-2\theta^2+3\theta+2}{\theta} p_U(\leq t^{\text{start}})$, then* $\text{abort}(t) \leq \theta\text{OPT}(\sigma_{\leq t})$ *for some $t > t^{\text{start}}$.*

(ii) *If requests are released at a time $t_j \geq t^{\text{start}} + 2p_U(\leq t^{\text{start}})$, then* $\text{abort}(t) \leq \theta\text{OPT}(\sigma_{\leq t})$ *for some $t > t^{\text{start}}$.*

(iii) *If requests are released at a time $t_j \geq t^{\text{start}} + p_U(\leq t^{\text{start}})$ with $p_U(= t_j) \geq p_U(\leq t^{\text{start}}) - (t_j - (t^{\text{start}} + p_U(\leq t^{\text{start}})))$, then* $\text{abort}(t) \leq \theta\text{OPT}(\sigma_{\leq t})$ *for some $t > t^{\text{start}}$.*

Proof. First of all let us assume that $p_U(= t) < p_U(\leq t^{\text{start}})$ for all $t > t^{\text{start}}$ since otherwise the statements are true by Lemma 3.14 as $\theta \geq 1 + \sqrt{2}/2$. Trivially, this also implies for all $t > t^{\text{start}}$ that

$$p^{\text{AOR}}(t) \leq p_U(\leq t^{\text{start}}). \tag{3.15}$$

Further, for each of the statements we suppose that $\text{abort}(t) > \theta\text{OPT}(\sigma_{\leq t})$ for all $t \in (t^{\text{start}}, t_j)$. Now, let us prove the three statements separately.

Proof of (i): Requests are released at time $t_j > t^{\text{start}}$ with $p_U(= t_j) \geq \frac{-2\theta^2+3\theta+2}{\theta} p_U(\leq t^{\text{start}})$. Since any tour for $\sigma_{\leq t_j}$ needs to return to the origin in the end, we obtain as a lower bound for the length of the optimal offline solution

$$\mathrm{OPT}(\sigma_{\le t_j}) \ge t_j + p_U(=t_j) \ge t_j + \frac{-2\theta^2+3\theta+2}{\theta} p_U(\le t^{\text{start}}). \tag{3.16}$$

As the server was located at the origin at time t^{start} we find for the release time t_j that

$$\begin{aligned} t_j \ge t^{\text{start}} + p^{\text{AOR}}(t_j) &\ge \theta \underbrace{\mathrm{OPT}(\sigma_{\le t^{\text{start}}})}_{\ge 2p_U(\le t^{\text{start}})} - 2p_U(\le t^{\text{start}}) + p^{\text{AOR}}(t_j) \\ &\ge (2\theta-2)p_U(\le t^{\text{start}}) + p^{\text{AOR}}(t_j). \end{aligned} \tag{3.17}$$

Using (3.16) and (3.17) we obtain for the ratio of $\mathrm{abort}(t_j)$ and $\mathrm{OPT}(\sigma_{\le t_j})$:

$$\begin{aligned} \frac{\mathrm{abort}(t_j)}{\mathrm{OPT}(\sigma_{\le t_j})} &\le \frac{t_j + p^{\text{AOR}}(t_j) + 2p_U(\le t^{\text{start}})}{t_j + \frac{-2\theta^2+3\theta+2}{\theta} p_U(\le t^{\text{start}})} && \text{by (3.16)} \\ &\le \frac{2\theta p_U(\le t^{\text{start}}) + 2p^{\text{AOR}}(t_j)}{\left(2\theta - 2 + \frac{-2\theta^2+3\theta+2}{\theta}\right) p_U(\le t^{\text{start}}) + p^{\text{AOR}}(t_j)} && \text{mon. decr. expr., (3.17)} \\ &\le \frac{(2\theta+2)p_U(\le t^{\text{start}})}{\left(2\theta - 1 + \frac{-2\theta^2+3\theta+2}{\theta}\right) p_U(\le t^{\text{start}})} && \text{mon. incr. expr., (3.15)} \\ &= \frac{2\theta+2}{2\theta - 1 + \frac{-2\theta^2+3\theta+2}{\theta}} = \theta. \end{aligned}$$

Proof of (ii): Requests are released at time $t_j \ge t^{\text{start}} + 2p_U(\le t^{\text{start}})$. By (i) we can assume that $p_U(\le t_j), p^{\text{AOR}}(t_j) \le \frac{-2\theta^2+3\theta+2}{\theta} p_U(\le t^{\text{start}})$. Thus, we have

$$\mathrm{abort}(t_j) \le t_j + 3\frac{-2\theta^2+3\theta+2}{\theta} p_U(\le t^{\text{start}})$$

which yields

$$\begin{aligned} \frac{\mathrm{abort}(t_j)}{\mathrm{OPT}(\sigma_{\le t_j})} &\le \frac{t_j + 3\frac{-2\theta^2+3\theta+2}{\theta} p_U(\le t^{\text{start}})}{t_j} \\ &\le 1 + \frac{3\frac{-2\theta^2+3\theta+2}{\theta} p_U(\le t^{\text{start}})}{2\theta p_U(\le t^{\text{start}})} \\ &= 1 + \frac{3\frac{-2\theta^2+3\theta+2}{\theta}}{2\theta} \le \theta \end{aligned}$$

where the last inequality holds only if $\theta \ge c$ with $c \approx 1.6961$ being the largest root of the polynomial $-X^3 - 2X^2 + 9/2X + 3$.

Proof of (iii): Requests are released at time $t_j \geq t^{\text{start}} + p_U(\leq t^{\text{start}})$ with $p_U(= t_j) \geq p_U(\leq t^{\text{start}}) - (t_j - (t^{\text{start}} + p_U(\leq t^{\text{start}})))$. By (i) we can assume that

$$p_U(= t) \leq \frac{-2\theta^2 + 3\theta + 2}{\theta} p_U(\leq t^{\text{start}}) \tag{3.18}$$

for all $t > t^{\text{start}}$. Consequently once the server reaches position $p_U(\leq t^{\text{start}})$ at time $t^{\text{start}} + p_U(\leq t^{\text{start}})$, it moves down at least until position $\frac{-2\theta^2+3\theta+2}{\theta} p_U(\leq t^{\text{start}})$ and then does not move above this position anymore. For $t = t_j$ in (3.18) we get

$$\frac{-2\theta^2 + 3\theta + 2}{\theta} p_U(\leq t^{\text{start}}) \geq p_U(\leq t^{\text{start}}) - (t_j - (t^{\text{start}} + p_U(\leq t^{\text{start}})))$$

which is equivalent to

$$t_j \geq t^{\text{start}} + p_U(\leq t^{\text{start}}) + \left(1 - \frac{-2\theta^2 + 3\theta + 2}{\theta}\right) p_U(\leq t^{\text{start}}).$$

Note that the right hand side of the last inequality is exactly the time it takes for the AOR-server to move to position $\frac{-2\theta^2+3\theta+2}{\theta} p_U(\leq t^{\text{start}})$ after having visited $p_U(\leq t^{\text{start}})$ at time $t^{\text{start}} + p_U(\leq t^{\text{start}})$. Hence, we obtain

$$p^{\text{AOR}}(t_j) \leq \frac{-2\theta^2 + 3\theta + 2}{\theta} p_U(\leq t^{\text{start}}).$$

Together with (3.18) for $t = t_j$ this yields for the length of the abort-schedule

$$\text{abort}(t_j) \leq t_j + 3\frac{-2\theta^2 + 3\theta + 2}{\theta} p_U(\leq t^{\text{start}}). \tag{3.19}$$

Further, for the optimal offline solution of $\sigma_{\leq t_j}$, we find that

$$\begin{aligned}
\text{OPT}(\sigma_{\leq t_j}) &\geq t_j + p_U(= t_j) \\
&\geq t_j + p_U(\leq t^{\text{start}}) - (t_j - (\underbrace{t^{\text{start}}}_{\geq \theta \text{OPT}(\sigma_{\leq t^{\text{start}}}) - 2p_U(\leq t^{\text{start}})} + p_U(\leq t^{\text{start}}))) \\
&\geq \theta \text{OPT}(\sigma_{\leq t^{\text{start}}}) \\
&\geq 2\theta p_U(\leq t^{\text{start}}).
\end{aligned} \tag{3.20}$$

Using (3.19) and (3.20) we obtain similarly to the proof of (ii) that

$$\begin{aligned}
\frac{\mathrm{abort}(t_j)}{\mathrm{OPT}(\sigma_{\leq t_j})} &\leq \frac{t_j + 3\frac{-2\theta^2+3\theta+2}{\theta} p_U(\leq t^{\mathrm{start}})}{\mathrm{OPT}(\sigma_{\leq t_j})} && \text{by (3.19)} \\
&\leq \underbrace{\frac{t_j}{\mathrm{OPT}(\sigma_{\leq t_j})}}_{\leq 1} + \frac{3\frac{-2\theta^2+3\theta+2}{\theta} p_U(\leq t^{\mathrm{start}})}{2\theta p_U(\leq t^{\mathrm{start}})} && \text{by (3.20)} \\
&\leq 1 + \frac{3\frac{-2\theta^2+3\theta+2}{\theta}}{2\theta} \leq \theta
\end{aligned}$$

for $\theta \geq c \approx 1.6961$. □

The following lemma specifies how the tour of the AOR-server back to the origin looks like once the server arrives at an uppermost position of a planned schedule.

Lemma 3.19. *Let $t, p_U \geq 0$ and $\sigma = \{(a_1, b_1; t_1), ..., (a_n, b_n; t_n)\}$ be a set of requests where $a_i < b_i \leq p_U$ and $t_i - (p_U - a_i) \leq t$ for all $i = 1, ..., n$. Then there is a unique optimal tour for σ starting at position p_U at time t of length $L^*(t, p_U, \sigma)$.*

Further, let $(a_{n+1}, b_{n+1}; t_{n+1})$ be a request with $a_{n+1} < b_{n+1} \leq p_U - (t_{n+1} - t)$, $t_{n+1} \in [0, t + p_U]$. Then the unique optimal tours for σ and $\sigma \cup \{(a_{n+1}, b_{n+1}; t_{n+1})\}$ starting at position p_U and at time t coincide until time t_{n+1}.

Proof. First of all note that any tour for σ starting at p_U at time t need not wait for request releases: As soon as the server reaches a position a_i, it can pick up the request $(a_i, b_i; t_i)$ since $t_i - (p_U - a_i) \leq t$ for all $i = 1, ..., n$. Further, we know that any tour for σ that starts at position p_U at time t needs to contain (possibly interrupted) segments where the server moves up from a_i to b_i for all $i \in \{1, ..., n\}$. Let $\{[a'_1, b'_1], ..., [a'_k, b'_k]\}$ be the set of intervals that is obtained by replacing intersecting intervals from $\{[a_1, b_1], ..., [a_n, b_n]\}$ with their unions (cf. Figure 3.7) such that

- for all $i \in \{1, ..., k\}$ there is a subset $I \subseteq \{1, ..., n\}$ with $[a'_i, b'_i] = \bigcup_{j \in I} [a_j, b_j]$
- and for all $i \in \{1, ..., k\}$, respectively $i \in \{1, ..., k-1\}$, we have $a'_i < b'_i$ and $b'_i < a'_{i+1}$.

Clearly, any tour starting at p_U at time t that serves σ needs to move up the intervals $[a'_i, b'_i]$ for all $i \in \{1, ..., k\}$ at some point of time. Thus, the unique optimal tour of length $L^*(t, p_U, \sigma)$ is given by

$$\mathrm{move}(a'_k) \oplus \mathrm{move}(b'_k) \oplus \cdots \oplus \mathrm{move}(a'_1) \oplus \mathrm{move}(b'_1) \oplus \mathrm{move}(0).$$

In order to see that the unique optimal tours for σ and $\sigma \cup \{(a_{n+1}, b_{n+1}; t_{n+1})\}$ starting at position p_U and at time t coincide until time t_{n+1}, let $a'_k, ..., a'_l$ be all a'_i with $a'_i \geq b_{n+1}$. The server of the optimal tour for $\sigma \cup \{(a_{n+1}, b_{n+1}; t_{n+1})\}$ as well as the one for the optimal tour for σ starts by traversing all the intervals $[a'_k, b'_k], ..., [a'_l, b'_l]$ and then moves downwards: The server of the optimal tour for $\sigma \cup \{(a_{n+1}, b_{n+1}; t_{n+1})\}$ moves towards some $a_i < b_{n+1}$ where $i \in \{1, ..., n+1\}$ depends on the exact values of a_{n+1} and b_{n+1} and the server of the optimal tour for σ moves towards $a'_{l-1} < b_{n+1}$. Consequently both tours coincide at least until they reach b_{n+1} which is not possible any earlier than t_{n+1} since $b_{n+1} \leq p_U - (t_{n+1} - t)$. □

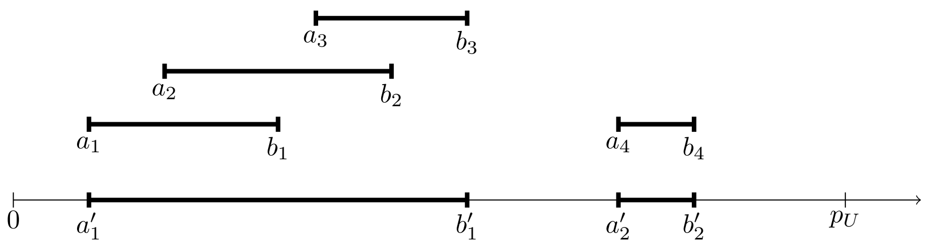

Figure 3.7: Example of intervals $\{[a_1, b_1], ..., [a_4, b_4]\}$ being replaced by $\{[a'_1, b'_1], [a'_2, b'_2]\}$.

Bibliography

1. Bjelde, A. *et al. Tight Bounds for Online TSP on the Line* in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (2017), 994–1005.
2. Lipmann, M. *On-line routing* PhD thesis (Technische Universiteit Eindhoven, 2003).
3. 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* (2000), 639–650.
4. Bonifaci, V., Lipmann, M. & Stougie, L. Online multi-server dial-a-ride problems. *Computational Complexity - CC* (2006).
5. Blom, M., Krumke, S., Paepe, de, W. & Stougie, L. *The online-TSP against fair adversaries* (Technische Universiteit Eindhoven, 1999).
6. Ausiello, G., Feuerstein, E., Leonardi, S., Stougie, L. & Talamo, M. Algorithms for the On-Line Travelling Salesman. *Algorithmica* **29,** 560–581 (2001).
7. Birx, A., Disser, Y. & Schewior, K. Improved Bounds for Open Online Dial-a-Ride on the Line. arXiv: `1907.02858` (2019).

Erklärung zur Abschlussarbeit gemäß §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Roman Edenhofer, die vorliegende Bachelor-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung überein.

Thesis Statement pursuant to §23 paragraph 7 of APB TU Darmstadt

I herewith formally declare that I, Roman Edenhofer, have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware, that in case of an attempt at deception based on plagiarism (§38 Abs. 2 APB), the thesis would be graded with 5,0 and counted as one failed examination attempt. The thesis may only be repeated once.

In the submitted thesis the written copies and the electronic version for archiving are identical in content.

Ort/Place, Datum/Date

Unterschrift des Autors/Signature of author