TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

The online-TSP against fair adversaries

Citation for published version (APA):
Blom, M., Krumke, S. O., Paepe, de, W. E., & Stougie, L. (1999). *The online-TSP against fair adversaries*. (Memorandum COSOR; Vol. 9923). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/1999

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 13. Feb. 2025

TU/e technische universiteit eindhoven
/department of mathematics and computing science

Memorandum COSOR 99-23

The Online-TSP Against Fair Adversaries

M. Blom, S.O. Krumke, W. de Paepe, L. Stougie

Eindhoven, December 1999
The Netherlands

The Online-TSP Against Fair Adversaries

Michiel Blom[1]*, Sven O. Krumke[2]**, Willem de Paepe[3], and Leen Stougie[4]***

[1] KPN-Research, P. O. Box 421, 2260AK Leidschendam, The Netherlands,
[2] Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, D-14195 Berlin-Dahlem, Germany,
krumke@zib.de
[3] Faculty of Technology Management, Technische Universiteit Eindhoven, P. O. Box 513, 5600MB Eindhoven, The Netherlands,
w.e.d.paepe@tm.tue.nl
[4] Faculty of Mathematics and Computer Science, Technische Universiteit Eindhoven, P. O. Box 513, 5600MB Eindhoven, The Netherlands,
L.Stougie@tue.nl

Abstract. In the online traveling salesman problem requests for visits to cities (points in a metric space) arrive online while the salesman is traveling. The salesman moves at no more than unit speed and starts and ends his work at a designated origin. The objective is to find a routing for the salesman which finishes as early as possible.
We consider the online traveling salesman problem when restricted to the non-negative part of the real line. We show that a very natural strategy is 3/2-competitive which matches our lower bound. The main contribution of the paper is the presentation of a "*fair adversary*", as an alternative to the omnipotent adversary used in competitive analysis for online routing problems. The fair adversary is required to remain inside the convex hull of the requests released so far. We show that on $\mathbb{R}_0^+$ algorithms can achieve a strictly better competitive ratio against a fair adversary than against a conventional adversary. Specifically, we present an algorithm against a fair adversary with competitive ratio $(1+\sqrt{17})/4 \approx 1.28$ and provide a matching lower bound. We also show competitiveness results for a special class of algorithms (called diligent algorithms) that do not allow waiting time for the server as long as there are requests unserved.

1 Introduction

The traveling salesman problem is a well studied problem in combinatorial optimization. In the classical setting, one assumes that the complete input of an instance is available for an algorithm to compute a solution. In many cases this

* Supported by the TMR Network DONET of the European Community ERB TMRX-CT98-0202
** Research supported by the German Science Foundation (DFG, grant Gr 883/5-2)
*** Supported by the TMR Network DONET of the European Community ERB TMRX-CT98-0202

offline optimization model does not reflect the real-world situation appropriately. In many applications not all requests for points to be visited are known in advance. Decisions have to be made *online* without knowledge about future requests.

Online algorithms are tailored to cope with such situations. Whereas offline algorithms work on the complete input sequence, online algorithms only see the requests released so far and thus, in planning a route, have to account for future requests that may or may not arise at a later time. A common way to evaluate the quality of online algorithms is *competitive analysis* [3, 5].

In this paper we consider the following online variant of the traveling salesman problem (called OLTSP in the sequel) which was introduced in [2]. Cities (requests) arrive online over time while the salesman is traveling. The requests are to be handled by a salesman-server that starts and ends his work at a designated origin. The objective is to find a routing for the server which finishes as early as possible (in scheduling theory this goal is usually referred to as minimizing the *makespan*). In this model it is feasible for the server to wait at the cost of time that elapses. Decisions are revocable, as long as they have not been executed. Only history is irrevocable.

Previous work. Ausiello et al. [2] present a 2-competitive algorithm for OLTSP which works in general metric spaces. The authors also show that for general metric spaces no deterministic algorithm can be c-competitive with $c < 2$. For the special case that the metric space is $\mathbb{R}$, the real line, they give a lower bound of $(9 + \sqrt{17})/8 \approx 1.64$ and a 7/4-competitive algorithm. Just very recently Lipmann [7] devised an algorithm that is best possible for this case with competitive ratio equal to the before mentioned lower bound.

Our contribution. In this paper the effect of restricting the class of algorithms allowed and restricting the power of the adversary in the competitive analysis is studied. We introduce and analyze a new class of online algorithms which we call *diligent algorithms*. Roughly speaking, a diligent algorithm never sits idle while there is work to do. A precise definition is presented in Section 3 where we also show that in general diligent algorithms are strictly weaker than algorithms that allow waiting time. In particular we show that no diligent algorithm can achieve a competitive ratio lower than 7/4 for the OLTSP on the real line. The 7/4-competitive algorithm in [2] is in fact a diligent algorithm and therefore best possible within this restricted class of algorithms.

We then concentrate on the special case of OLTSP when the underlying metric space is $\mathbb{R}_0^+$, the non-negative part of the real line. In Section 4 we show that an extremely simple and natural diligent strategy is 3/2-competitive and that this result is best possible for (diligent and non-diligent) deterministic algorithms on $\mathbb{R}_0^+$. The main contribution is contained in Section 5. Here we deal with an objection frequently encountered against competitive analysis concerning the unrealistic power of the adversary against which performance is measured. Indeed, in the OLTSP on the real line the before mentioned 7/4-competitive algorithm

reaches its competitive ratio against an adversary that moves away from the previously released requests without giving any information to the online algorithm. We introduce a *fair adversary* that is in a natural way restricted in the context of the online traveling salesman problem studied here. It should be seen as a more reasonable adversary model. A fair adversary always keeps its server within the convex hull of the requests released so far. We show that this adversary model indeed allows for lower competitive ratios. For instance, the above mentioned 3/2-competitive diligent strategy against the conventional adversary is 4/3-competitive against the fair adversary. This result is best possible for diligent algorithms against a fair adversary.

	Diligent Algorithms	General Algorithms
General Adversary	LB = UB = 3/2	LB = UB = 3/2
Fair Adversary	LB = UB = 4/3	LB = UB = $(1+\sqrt{17})/4$

Table 1. Overview of the lower bound (LB) and upper bound (UB) results for the competitive ratio of deterministic algorithms for OLTSP on $\mathbb{R}_0^+$ in this paper.

We also present a non-diligent algorithm with competitive ratio $(1+\sqrt{17})/4 \approx 1.28 < 4/3$ competing against the fair adversary. Our result is the first one that shows that waiting is actually advantageous in the OLTSP. The before mentioned algorithm in [7] also uses waiting, but became known after the one presented in this paper and has not been officially published yet. Such results are known already for online scheduling problems (see e.g. [6,4,8]) and, again very recently, also for an online dial-a-ride problem [1]. Our competitiveness result is complemented by a matching lower bound on the competitive ratio of algorithms against the fair adversary. Table 1 summarizes our results for OLTSP on $\mathbb{R}_0^+$.

2 Preliminaries

An instance of the *Online Traveling Salesman Problem* (OLTSP) consists of a metric space $M = (X, d)$ with a distinguished origin $o \in M$ and a sequence $\sigma = \sigma_1, \ldots, \sigma_m$ of requests. In this paper we are mainly concerned with the special case that M is $\mathbb{R}_0^+$, the non-negative part of the real line endowed with the Euclidean metric, i.e., $X = \mathbb{R}_0^+ = \{ x \in \mathbb{R} : x \geq 0 \}$ and $d(x, y) = |x - y|$. A server is located at the origin o at time 0 and can move at most at unit speed.

Each *request* is a pair $\sigma_i = (t_i, x_i)$, where $t_i \in \mathbb{R}$ is the time at which request σ_i is released (becomes known), and $x_i \in X$ is the point in the metric space requested to be visited. We assume that the sequence $\sigma = \sigma_1, \ldots, \sigma_m$ of requests is given in order of non-decreasing release times. For a real number t we denote by $\sigma_{\leq t}$ and $\sigma_{<t}$ the subsequence of requests in σ released up to time t and strictly before time t, respectively.

It is assumed that the online algorithm does neither have information about the time when the last request is released nor about the total number of requests.

An online algorithm for OLTSP must determine the behavior of the server at a certain moment t of time as a function of all the requests in $\sigma_{\leq t}$ (and the current time t). In contrast, the offline algorithm has information about all requests in the whole sequence σ already at time 0. A feasible online/offline solution is a route for the server which serves all requested points, where each request is served not earlier than the time it is released, and which starts and ends in the origin o.

The objective in the OLTSP is to minimize the total completion time (also called the "makespan" in scheduling) of the server, that is, the time when the server has served all requests and returned to the origin.

Let $\text{ALG}(\sigma)$ denote the completion time of the server moved by algorithm ALG on the sequence σ of requests. We use OPT to denote the optimal offline algorithm. An online algorithm ALG for OLTSP is *c-competitive*, if there exists a constant c such that for every request sequence σ the inequality $\text{ALG}(\sigma) \leq c \cdot \text{OPT}(\sigma)$ holds.

3 Diligent algorithms

In this section we introduce a particular class of algorithms for OLTSP which we call *diligent algorithms*. Intuitively, a diligent algorithm should never sit and wait when it could serve yet unserved requests. A diligent server should also move towards work that has to be done directly without any detours. To translate this intuition into a rigorous definition some care has to be taken.

Definition 1 (Diligent Algorithm). *An algorithm* ALG *for* OLTSP *is called* diligent, *if it satisfies the following conditions:*

1. *If there are still unserved requests, then the direction of the server operated by* ALG *changes only if a new request becomes known, or the server is either in the origin or at a request that has just been served.*
2. *At any time when there are yet unserved requests, the server operated by* ALG *either moves towards an unserved request or the origin at maximum (i.e. unit) speed. The latter case is only allowed if the server operated by* ALG *is not yet in the origin.*

We emphasize that a diligent algorithm is allowed to move its server towards an unserved request and change his direction towards another unserved request or to the origin at the moment a new request becomes known.

Lemma 1. *No diligent online algorithm for* OLTSP *on the real line* $\mathbb{R}$ *has competitive ratio of less than* $7/4$.

Proof. Suppose that ALG is a diligent algorithm for OLTSP on the real line. Consider the following adversarial input sequence. At time 0 two requests $\sigma_1 = (0, 1/2)$ and $\sigma_2 = (1/2, 0)$ are released. There will be no further requests before

time 1. Thus, by the diligence of the algorithm the server will be at the origin at time 1.

At time 1 two new requests at points 1 and -1, respectively, are released. Since the algorithm is diligent, starting at time 1 it must move its server to one of these requests at maximum, i.e., unit, speed. Without loss of generality assume that this is the request at 1. ALG's server will reach this point at time 2. Starting at time 2, ALG will have to move its server either directly towards the unserved request at -1 or towards the origin, which essentially gives the same movement and implies that the server is at the origin at time 3. At that time, the adversary issues another request at 1. Thus, ALG's server will still need at least 4 time units to serve -1 and 1 and return at the origin. Therefore, he will not be able to complete its work before time 7.

The adversary handles the sequence by first serving the request at -1, then the two requests at 1 and finally returns to the origin at time 4, yielding the desired result. □

This lower bound shows that the 7/4-competitive algorithm presented in [2], which is in fact a diligent algorithm is best possible within the class of diligent algorithms for the OLTSP on the real line.

4 The Oltsp on the non-negative part of the real line

We first consider OLTSP on $\mathbb{R}_0^+$ when the offline adversary is the conventional (omnipotent) opponent.

Theorem 1. *No deterministic algorithm for* OLTSP *on* $\mathbb{R}_0^+$ *has a competitive ratio of less than* 3/2.

Proof. At time 0 the request $\sigma_1 = (0, 1)$ is released. Let T be the time that the server operated by ALG has served the request σ_1 and returned to the origin 0. If $T \geq 3$, then no further request is released and ALG is no better than 3/2-competitive since $\text{OPT}(\sigma_1) = 2$. Thus, assume that $T < 3$. In this case the adversary releases a new request $\sigma_2 = (T, T)$. Clearly, $\text{OPT}(\sigma_1, \sigma_2) = 2T$. On the other hand $\text{ALG}(\sigma_1, \sigma_2) \geq 3T$, yielding a competitive ratio of 3/2. □

The following extremely simple strategy achieves a competitive ratio that matches this lower bound (as we will show below):

Strategy mrin("Move-Right-If-Necessary") If a new request is released and the request is to the right of the current position of the server operated by MRIN, then the MRIN-server starts to move right at full speed. The server continues to move right as long as there are yet unserved requests to the right of the server. If there are no more unserved requests to the right, then the server moves towards the origin 0 at full speed.

It is easy to verify that Algorithm MRIN is in fact a diligent algorithm. The following theorem shows that the strategy has a best possible competitive ratio for OLTSP on $\mathbb{R}_0^+$.

Theorem 2. *Strategy* MRIN *is a diligent* 3/2*-competitive algorithm for the* OLTSP *on the non-negative part* $\mathbb{R}_0^+$ *of the real line.*

Proof. We show the theorem by induction on the number of requests in the sequence σ. It clearly holds if σ contains at most one request. The induction hypothesis is that it holds for any sequence of $m-1$ requests.

Suppose that request $\sigma_m = (t, x)$ is the last request of $\sigma = \sigma_1, \ldots, \sigma_{m-1}, \sigma_m$. If $t = 0$, then MRIN is obviously 3/2-competitive, so we will assume that $t > 0$. Let f be the position of the request unserved by the MRIN-server at time t (excluding σ_m), which is furthest away from the origin.

In case $x \leq f$, MRIN's cost for serving σ is equal to the cost for serving the sequence consisting of the first $m-1$ requests of σ. Since new requests can never decrease the optimal offline cost, the induction hypothesis implies the theorem.

Now assume that $f < x$. Thus, at time t the request in x is the furthest unserved request. MRIN will complete its work no later than time $t + 2x$. The optimal offline cost $\text{OPT}(\sigma)$ is bounded from below by $\max\{t+x, 2x\}$. Therefore,

$$\frac{\text{MRIN}(\sigma)}{\text{OPT}(\sigma)} \leq \frac{t+x}{\text{OPT}(\sigma)} + \frac{x}{\text{OPT}(\sigma)} \leq \frac{t+x}{t+x} + \frac{x}{2x} = \frac{3}{2}.$$

□

The result established above can be used to obtain competitiveness results for the situation of the OLTSP on the real line when there are more than one server, and the goal is to minimize the time when the last of its servers returns to the origin 0 after all requests have been served.

Lemma 2. *There is an optimal offline strategy for* OLTSP *on the real line with* $k \geq 2$ *servers such that no server ever crosses the origin.*

Proof. Omitted in this abstract. □

Corollary 1. *There is a* 3/2*-competitive algorithm for the* OLTSP *with* $k \geq 2$ *servers on the real line.* □

5 Fair Adversaries

The adversaries used in the bounds of the previous section are abusing their power in the sense that they can move to points where they know a request will pop up without revealing the request to the online server before reaching the point. As an alternative we propose the following more reasonable adversary that we baptized *fair adversary*. We show that we can obtain better competitive ratios for the OLTSP on $\mathbb{R}_0^+$ under this model. We will also see that under this adversary model there does exist a distinction in competitiveness between diligent and non-diligent algorithms. Recall that $\sigma_{<t}$ is the subsequence of σ consisting of those requests with release time strictly smaller than t.

Definition 2 (Fair Adversary). *An offline adversary for the* OLTSP *in the Euclidean space* $(\mathbb{R}^n, \|.\|)$ *is* fair, *if at any moment* t, *the position of the server operated by the adversary is within the convex hull of the origin* o *and the requested points from* $\sigma_{<t}$.

In the special case of $\mathbb{R}_0^+$ a fair adversary must always keep its server in the interval $[0, F]$, where F is the position of the request with the largest distance to the origin 0 among all requests released so far. The following lower bound result shows that the OLTSP on the real line against fair adversaries is still a non-trivial problem.

Theorem 3. *No deterministic algorithm for* OLTSP *on* $\mathbb{R}$ *has competitive ratio less than* $(5+\sqrt{57})/8 \approx 1.57$ *against a fair adversary.*

Proof. Suppose that there exists a c-competitive online algorithm. The adversarial sequence starts with two requests at time 0, $\sigma_1 = (0, 1)$ and $\sigma_2 = (0, -1)$. Without loss of generality, we suppose that the first request that is served is σ_1. At time 2 the online server can't have served both requests. We distinguish two main cases divided in some sub-cases.
Case 1: *None of the requests has been served at time* 2.

- If at time 3 request σ_1 is still unserved, let t' be the first time the server crosses the origin after serving the request. Clearly, $t' \geq 4$. At time t' the online server still has to visit the request in -1. If $t' > 4c-2$ the server can not be c-competitive because the fair adversary can finish the sequence at time 4.
  Thus, suppose that $4 \leq t' \leq 4c-2$. At time t' a new request $\sigma_3 = (t', 1)$ is released. The online server can not finish the complete sequence before $t'+4$, whereas the adversary needs at $t'+1$. Therefore, $c \geq \frac{t'+4}{t'+1}$ and for $4 \leq t' \leq 4c-2$ we have that

$$c \geq \frac{(4c-2)+4}{(4c-2)+1} = \frac{4c+2}{4c-1}$$

  implying $c \geq (5+\sqrt{57})/8$.
- If at time 3 the request σ_1 has already been served, the online server can not be to the left of the origin at time 3 (given the fact that at time 2 no request had been served). The adversary now gives a new request $\sigma_3 = (3, 1)$. There are two possibilities: either σ_2, the request in -1, is served before σ_3 or the other way round.
  If the server decides to serve σ_2 before σ_3 then it can not complete before time 7. Since the adversary completes the sequence in time 4, the competitive ratio is at least $7/4$.
  If the online server serves σ_3 first, then again, let t' be the time that the server crosses the origin after serving σ_3. As before, we must have $4 \leq t' \leq 4c-2$. At time t' the fourth request $\sigma_4 = (t', 1)$ is released. The same arguments as above apply to show that the algorithm is at least $(5+\sqrt{57})/8$-competitive.

Case 2: *One of the requests has been served at time 2 by the online server.* We assume without loss of generality that σ_1 has been served. At time 2 the third request $\sigma_3 = (2,1)$ is released. In fact, we are back in the situation in which at time 2 none of the two requests are served. In case the movements of the online server are such that no further request is released by the adversary, the latter will complete at time 4. In the other cases the last released requests are released after time 4 and the adversary can still reach them in time. □

For comparison, the lower bound on the competitive ratio for the OLTSP in $\mathbb{R}$ against an adversary that is not restricted to be fair is $(9+\sqrt{17})/8$ [2]. As mentioned before, only recently Lipmann [7] presented a $(9+\sqrt{17})/8$-competitive algorithm against a non-fair adversary. He conjectures that a similar type of algorithm will also be best possible against a fair adversary. In contrast, the picture for the problem on the non-negative part of the real line is already complete (see Theorems 5 and 6 for diligent algorithms and Theorems 4 and 7 for non-diligent algorithms below).

Theorem 4. *No deterministic algorithm for* OLTSP *on* $\mathbb{R}_{\geq 0}$ *has competitive ratio of less than* $(1+\sqrt{17})/4 \approx 1.28$ *against a fair adversary.*

Proof. Suppose that ALG is c-competitive. At time 0 the adversary releases request $\sigma_1 = (0,1)$. Let T denote the time that the server operated by ALG has served this request and is back at the origin. For ALG to be c-competitive, we must have that $T \leq c \cdot \text{OPT}(\sigma_1) = 2c$, otherwise no further requests will be released. At time T the adversary releases a second request $\sigma_2 = (T,1)$. The completion time of ALG becomes then at least $T+2$.

On the other hand, starting at time 0 the fair adversary moves its server to 1, lets it wait there until time T and then goes back to the origin 0 yielding a completion time of $T+1$. Therefore,

$$\frac{\text{ALG}(\sigma)}{\text{OPT}(\sigma)} \geq \frac{T+2}{T+1} \geq \frac{2c+2}{2c+1} = 1 + \frac{1}{2c+1},$$

given the fact that $T \leq 2c$. Since by assumption ALG is c-competitive, we have that $1 + 1/(2c+1) \leq c$, implying that $c \geq (1+\sqrt{17})/4$. □

For diligent algorithms we can show a higher lower bound against a fair adversary.

Theorem 5. *No deterministic diligent algorithm for* OLTSP *on* $\mathbb{R}_0^+$ *has competitive ratio of less than* $4/3$ *against a fair adversary.*

Proof. Consider the adversarial sequence $\sigma_1 = (0,1)$, $\sigma_2 = (1,0)$, and $\sigma_3 = (2,1)$. By its diligence the online algorithm will start to travel to 1 at time 0, back to 0 at time 1, arriving there at time 2. Then its server has to visit 1 again, so that he will finish no earlier than time 4. Obviously, the optimal offline solution is to leave 1 not before time 2, and finishing at time 3. □

We show now that the algorithm MRIN presented before has a better competitive ratio against the fair adversary than the ratio of 3/2 achieved against a conventional adversary. In fact we show that the ratio matches the lower bound for diligent algorithms proved in the previous theorem.

Theorem 6. *Strategy* MRIN *is a* $4/3$*-competitive algorithm for the* OLTSP *on* $\mathbb{R}_0^+$ *against a fair adversary.*

Proof. Omitted in this abstract.

Thus, Algorithm MRIN attains a best possible competitive ratio against the fair adversary among all diligent algorithms. Given the lower bound for general non-diligent algorithms in Theorem 4 we aim now at designing an online algorithm that obtains better competitive ratios against a fair adversary. In view of Theorem 5 such an algorithm will have to be non-diligent, i.e., incorporate waiting times.

The problem with Algorithm MRIN is that shortly after it starts to return towards the origin from the furthest previously unserved request, a new request to the right of its server arrives. In this case the MRIN-server has to return to a position it just left. Algorithm WS presented below attempts successfully to avoid this pitfall.

Strategy ws("Wait Smartly") The WS-server moves right if there are yet unserved requests to the right of his present position. Otherwise, it takes the following actions. Suppose it arrives at his present position, which is a currently rightmost unserved request, $s(t)$ at time t.

1. Compute the the optimal offline solution value $\text{OPT}(\sigma_{\leq t})$ for all requests released up to time t.
2. Determine a waiting time $W := \alpha \cdot \text{OPT}(\sigma_{\leq t}) - s(t) - t$, with $\alpha = (1+\sqrt{17})/4$.
3. Wait at point $s(t)$ until time $t + W$ and then start to move back to the origin 0.

We notice that when the server is moving back to the origin and no new requests are released until time $t + W + s(t)$, then the WS-server reaches the origin 0 at time $t + W + s(t) = \alpha \cdot \text{OPT}(\sigma_{\leq t})$ having served all requests released so far. If a new request is released at time $t' \leq W + t + s(t)$ and the request is to the right of $s(t')$, then the WS-server starts to move to the right immediately until it reaches the furthest unserved request.

Theorem 7. *Algorithm* WS *is* α*-competitive with* $\alpha = (1+\sqrt{17})/4 \approx 1.28$ *for the* OLTSP *on* $\mathbb{R}_0^+$ *against a fair adversary.*

Proof. By the definition of the waiting time it is sufficient to prove that at any point where a waiting time is computed this waiting time is non-negative. In that case the server will always return at o before time $\alpha\,\text{OPT}(\sigma)$. This is clearly true if the sequence σ contains at most one request. We make the induction hypothesis that it is also true for any sequence of at most $m-1$ requests.

Let $\sigma = \sigma_1, \ldots, \sigma_m$ be any sequence of requests and let $\sigma_m =: (t, x)$ be the request released last. If $t = 0$, then there is nothing left to show, so we will assume for the remainder of the proof that $t > 0$.

We denote by $s(t)$ and $s^*(t)$ the positions of the WS- and the fair adversary's server at time t, respectively. We also let $r_f = (t_f, f)$ be the furthest (i.e. most remote from the origin) yet unserved request by WS at time t excluding the request σ_m. Finally, let $r_F = (t_F, F)$ be the furthest released request in $\sigma_1, \ldots, \sigma_{m-1}$. Obviously $f \leq F$. Again, we distinguish three different cases depending on the position of x relative to f and F.

Case 1: $x \leq f$

Since the WS-server has to travel to f anyway and by the induction hypothesis there was a non-negative waiting time in f or $s(t)$ (depending on whether $s(t) > f$ or $s(t) \leq f$) before request σ_m was released, the waiting time in f or $s(t)$ can not decrease since the optimal offline completion time can not decrease by an additional request).

Case 2: $f \leq x < F$

If $s(t) \geq x$, then again by the induction hypothesis and the fact that the route length of WS's server does not increase, the possible waiting time at $s(t)$ is non-negative.

Thus we can assume that $s(t) < x$. The WS-server will now travel to point x, arrive there at time $t + d(s(t), x)$, and possibly wait there some time W before returning to the origin, with

$$W = \alpha \, \mathrm{OPT}(\sigma) - (t + d(s(t), x)) - x.$$

Inserting the obvious lower bound $\mathrm{OPT}(\sigma) \geq t + x$ yields

$$W \geq (\alpha - 1)\mathrm{OPT}(\sigma) - d(s(t), x). \tag{1}$$

To bound $\mathrm{OPT}(\sigma)$ in terms of $d(s(t), x)$ consider the time t' when the WS-server had served the request at F and started to move left. Clearly $t' < t$ since otherwise $s(t)$ could not be smaller than x as assumed. Thus, the subsequence $\sigma_{\leq t'}$ of σ does not contain (t, x). By the induction hypothesis, WS is α-competitive for the sequence $\sigma_{\leq t'}$. At time t' when he left F he would have arrived in the origin at time $\alpha\mathrm{OPT}(\sigma_{\leq t'})$, i.e.,

$$t' + F = \alpha \cdot \mathrm{OPT}(\sigma_{\leq t'}). \tag{2}$$

Notice that $t \geq t' + d(F, s(t))$. Since $\mathrm{OPT}(\sigma_{\leq t'}) \geq 2F$ we obtain from (2) that

$$t \geq \alpha 2F - F + d(F, s(t)) = (2\alpha - 1)F + d(s(t), F). \tag{3}$$

Since by assumption we have $s(t) < x < F$ we get that $d(s(t), x) \leq d(s(t), F)$ and $d(s(t), x) \leq F$, which inserted in (3) yields

$$t \geq (2\alpha - 1)d(s(t), x) + d(s(t), x) = 2\alpha d(s(t), x). \tag{4}$$

We combine this with the previously mentioned lower bound $\mathrm{OPT}(\sigma) \geq t + x$ to obtain:

$$\mathrm{OPT}(\sigma) \geq 2\alpha d(s(t), x) + x \geq (2\alpha + 1)d(s(t), x). \tag{5}$$

Using inequality (5) in (1) gives

$$\begin{aligned} W &\geq (\alpha-1)(2\alpha+1)d(s(t),x) - d(s(t),x) \\ &= (2\alpha^2-\alpha-2)d(s(t),x) \\ &= \left(\frac{9+\sqrt{17}}{4} - \frac{1+\sqrt{17}}{4} - 2\right) d(s(t),x) \\ &= 0. \end{aligned}$$

This completes the proof for the second case.

Case 3: $f \leq F \leq x$

Starting at time t the WS-server moves to the right until he reaches x, and after waiting there an amount W returns to 0, with

$$W = \alpha\,\mathrm{OPT}(\sigma) - (t + d(s(t),x)) - x. \tag{6}$$

We will show that also in this case $W \geq 0$. At time t the adversary's server still has to travel at least $d(s^*(t),x) + x$ units. This results in

$$\mathrm{OPT}(\sigma) \geq t + d(s^*(t),x) + x.$$

Since the offline adversary is fair, its position $s^*(t)$ at time t can not be strictly to the right of F.

$$\mathrm{OPT}(\sigma) \geq t + d(F,x) + x. \tag{7}$$

Insertion into (6) yields

$$W \geq (\alpha-1)\mathrm{OPT}(\sigma) - d(s(t),F) \tag{8}$$

since $F > s(t)$ by definition of the algorithm.

The rest of the arguments are similar to those used in the previous case. Again that WS's server started to move to the left from F at some time $t' < t$, and we have

$$t' + F = \alpha \cdot \mathrm{OPT}(\sigma_{\leq t'}). \tag{9}$$

Since $t \geq t' + d(s(t),F)$ and $\mathrm{OPT}(\sigma_{\leq t'}) \geq 2F$ we obtain from (9) that

$$t \geq \alpha 2F - F + d(s(t),F) = (2\alpha-1)F + d(s(t),F) \geq 2\alpha d(s(t),F).$$

We combine this with (7) and the fact that $x \geq d(s(t),F)$ to achieve

$$\mathrm{OPT}(\sigma) \geq 2\alpha d(s(t),F) + d(F,x) + x \geq (2\alpha+1)d(s(t),F).$$

Using this inequality in (8) gives

$$\begin{aligned} W &\geq (\alpha-1)(2\alpha+1)d(s(t),F) - d(s(t),F) \\ &= (2\alpha^2-\alpha-2)d(s(t),F) \\ &= \left(\frac{9+\sqrt{17}}{4} - \frac{1+\sqrt{17}}{4} - 2\right) d(s(t),F) \\ &= 0. \end{aligned}$$

This completes the proof. □

6 Conclusions

We introduced an alternative more fair performance measure for online algorithms for the traveling salesman problem. The first results are encouraging. On the non-negative part of the real line the fair model allows a strictly lower competitive ratio than the conventional model with an omnipotent adversary.

Next to that we considered a restricted class of algorithms for the online traveling salesman problems, suggestively called diligent algorithms. We showed that in general diligent algorithms have strictly higher competitive ratios than algorithms that sometimes leaves the server idle, to wait for possible additional information. In online routing companies, like courier services or transportation companies waiting instead of immediately starting as soon as requests are presented is common practice. Our results support this strategy.

It is still open to find a best possible non-diligent algorithm for the problem on the real line against a fair adversary. However, it is very likely that an algorithm similar to the best possible algorithm presented in [7] against a non-fair adversary will appear to be best possible for this case.

We notice here that for general metric spaces the lower bound of 2 on the competitive ratio of algorithms in [2] is established with a fair adversary as opponent. Moreover, a diligent algorithm is presented which has a competitive ratio that meets the lower bound.

We hope to have encouraged research into ways to restrict the power of adversaries in online competitive analysis.

Acknowledgement: Thanks to Maarten Lipmann for providing the lower bound in Theorem 3.

References

1. N. Ascheuer, S. O. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In *Proceedings of the 17th International Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, 2000. To appear.
2. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line traveling salesman. *Algorithmica*, 1999. To appear.
3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
4. B. Chen, A. P. A. Vestjens, and G. J. Woeginger. On-line scheduling of two-machine open shops where jobs arrive over time. *Journal of Combinatorial Optimization*, 1:355–365, 1997.
5. A. Fiat and G. J. Woeginger, editors. *Online Algorithms: The State of the Art*, volume 1442 of *Lecture Notes in Computer Science.* Springer, 1998.
6. J. A. Hoogeveen and A. P. A. Vestjens. Optimal on-line algorithms for single-machine scheduling. In *Proceedings of the 5th Mathematical Programming Society Conference on Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, pages 404–414, 1996.

7. M. Lipmann. The online traveling salesman problem on the line. Master's thesis, Department of Operations Research, University of Amsterdam, The Netherlands, 1999.
8. C. Philips, C. Stein, and J. Wein. Minimizing average completion time in the presence of release dates. In *Proceedings of the 4th Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 86–97, 1995.