ELSEVIER

Theoretical Computer Science 295 (2003) 279–294

Theoretical Computer Science

www.elsevier.com/locate/tcs

News from the online traveling repairman

Sven O. Krumke[a,*,1], Willem E. de Paepe[b], Diana Poensgen[a], Leen Stougie[c,d,2]

[a] *Konrad-Zuse-Zentrum für Informationstechnik Berlin, Department Optimization, Takustr. 7, 0-14195 Berlin-Dahlem, Germany*

[b] *Department of Technology Management, Technical University of Eindhoven, P.O. Box 513, 5600MB Eindhoven, The Netherlands*

[c] *Department of Mathematics, Technical University of Eindhoven, P.O. Box 513, 5600MB Eindhoven, The Netherlands*

[d] *Centre for Mathematics and Computer Science (CWI), P.O. Box 94079, NL-1090 GB Amsterdam, The Netherlands*

Received 26 September 2001; received in revised form 13 December 2001; accepted 27 December 2001

Abstract

In the traveling repairman problem (TRP), a tour must be found through every one of a set of points (cities) in some metric space such that the weighted sum of completion times of the cities is minimized. Given a tour, the completion time of a city is the time traveled on the tour before the city is reached. In the online traveling repairman problem (OLTRP) requests for visits to cities arrive online while the repairman is traveling. We analyze the performance of algorithms for the online problem using competitive analysis, where the cost of an online algorithm is compared to that of an optimal offline algorithm.

We show how to use techniques from online-scheduling to obtain a deterministic algorithm with a competitive ratio of $(1+\sqrt{2})^2 < 5.8285$ for the OLTRP in general metric spaces. We also present a randomized algorithm which achieves a competitive ratio of $4/\ln 3 < 3.6410$ against an oblivious adversary. Our results extend to the "dial-a-ride" generalization L-OLDARP of the OLTRP, where objects have to be picked up and delivered by a server. This improves upon the previously best competitive ratio of 9 for the OLTRP on the real line and, moreover, the results are valid for any metric space. For the case of the L-OLDARP our algorithms are the first competitive algorithms.

We also derive the first lower bounds for the competitive ratio of randomized algorithms for the OLTRP and the L-OLDARP against an oblivious adversary. Our lower bounds are $(\ln 16+1)/$

* Corresponding author.

E-mail addresses: krumke@zib.de (S.O. Krumke), w.e.d.paepe@tm.tue.nl (W.E. de Paepe), poensgen@zib.de (D. Poensgen), leen@win.tue.nl (L. Stougie).

[1] Research supported by the German Science Foundation (DFG, grant Gr 883/10).

[2] Supported by the TMR Network DONET of the European Community ERB TMRX-CT98-0202.

PII: S0304-3975(02)00409-7

$(\ln 16-1) > 2.1282$ for the L-OLDARP on the line, $(4e-5)/(2e-3) > 2.41041$ for the L-OLDARP on general metric spaces, 2 for the OLTRP on the line, and $\frac{7}{3}$ for the OLTRP on general metric spaces.

Keywords: Online Algorithms; Competitive analysis; Traveling repairman; Latency; Dial-a-ride problem

1. Introduction

In the *traveling repairman problem* (TRP) [1] a server must visit a set of m points $p_1,\ldots,p_m$ in a metric space. The server starts in a designated point 0 of the metric space, called the *origin*, and travels at most at unit speed. Given a tour through the m points, the completion time C_j of point p_j is defined as the time traveled by the server on the tour until it reaches p_j $(j=1,\ldots,m)$. Each point p_j has a weight w_j, and the objective of the TRP is to find the tour that minimizes the total weighted completion time $\sum_{j=1}^{m} w_j C_j$. This objective is also referred to as the *latency*.

In this paper we consider an online version of the TRP called the *online traveling repairman problem* (OLTRP). Requests for visits to points are released over time while the repairman (the server) is traveling. In the online setting the completion time of a request r_j at point p_j with release time t_j is the first time at which the repairman visits p_j after the release time t_j. The online model allows the server to wait. However, waiting yields an increase in the completion times of the points still to be served. Decisions are revocable as long as they have not been executed.

In the dial-a-ride generalization L-OLDARP (for "latency online dial-a-ride problem") each request specifies a ride from one point in the metric space, its *source*, to another point, its *destination*. The server can serve only one ride at a time, and preemption of rides is not allowed: once a ride is started it has to be finished without interruption.

An online algorithm does not know about the existence of a request before its release time. It must base its decisions solely on past information. A common way to evaluate the quality of online algorithms is *competitive analysis* [5]: An algorithm is called *c-competitive* if its cost on any input sequence is at most c times the cost of an optimal offline algorithm.

1.1. Related work

Feuerstein and Stougie [7] presented a 9-competitive algorithm for the OLTRP on the real line and a 15-competitive algorithm for the L-OLDARP on the real line for the special case that the server has infinite capacity. In the same paper lower bounds of $1+\sqrt{2}$ and 3 on the competitive ratio of any deterministic algorithm for the OLTRP and the L-OLDARP, respectively, are proved.

The offline traveling repairman problem TRP is known to be NP-hard [1]. In the special case when the metric space is the real line, the TRP can be solved in polynomial time [1]. However, up to now, the complexity of the TRP on weighted trees remains unsettled. Approximation algorithms for the TRP have been studied in [2,4,8].

Table 1
Deterministic upper and lower bounds for the OLTRP and the L-OLDARP

	Deterministic UB	Previous best UB	Deterministic LB
OLTRP	$(1+\sqrt{2})^2 < 5.8285$ (Corollary 8)	9, see [7] (real line)	$1+\sqrt{2}$, see [7]
L-OLDARP	$(1+\sqrt{2})^2 < 5.8285$ (Theorem 7)	15, see [7] (real line, server capac. ∞)	3, see [7]

Table 2
Randomized upper and lower bounds for the OLTRP and the L-OLDARP

	Randomized UB	Randomized LB
OLTRP	$\frac{4}{\ln 3} < 3.6410$ (Corollary 12)	general: $\frac{7}{3}$ (Theorem 16) real line: 2 (Theorem 17)
L-OLDARP	$\frac{4}{\ln 3} < 3.6410$ (Theorem 11)	general: $\frac{4e-5}{2e-3} > 2.4104$ (Theorem 14) real line: $\frac{\ln 16+1}{\ln 16-1} > 2.1282$ (Theorem 15)

1.2. Contribution and paper outline

Our main results are competitive algorithms and randomized lower bounds for the OLTRP and the L-OLDARP. Our algorithms improve the competitive ratios given in [7], and, moreover, the results are valid for any metric space and not just the real line. For the case of the L-OLDARP our algorithms are the first competitive algorithms. The randomized lower bounds are the first ones for the OLTRP and the L-OLDARP.

Our algorithms are adaptations of the GREEDY-INTERVAL algorithm for online scheduling presented in [9,10] and of the randomized version given in [6]. Our lower bound results are obtained by applying Yao's principle in conjunction with a technique of Seiden [12]. An overview of the results is given in Tables 1 and 2.

In Section 2 we define the problems formally. Section 3 contains the deterministic algorithm INTERVAL$_\alpha$ and the proof of its competitive ratio. In Section 4 we present the randomized algorithm RANDINTERVAL$_\alpha$ which achieves a better competitive ratio. In Section 5 we prove lower bounds on the competitive ratio of randomized algorithms against an oblivious adversary.

2. Preliminaries

An instance of the online dial-a-ride problem OLDARP consists of a metric space $M=(X,d)$ with a distinguished origin $o \in X$ and a sequence $\sigma = r_1, \ldots, r_m$ of requests.

We assume that M has the property that for all pairs of points $(x, y) \in M$ there is a continuous path $p : [0, 1] \to X$ in X with $p(0) = x$ and $p(1) = y$ of length $d(x, y)$ (see [3] for a thorough discussion of this model). Examples of metric spaces that satisfy the above condition are the Euclidean space $\mathbb{R}^p$ and a metric space induced by a connected undirected edge-weighted graph.

Each request $r_j = (t_j, a_j, b_j, w_j)$ specifies a ride, defined by two points: $a_j \in X$, the ride's *source*, and $b_j \in X$, its *destination*, and a weight $w_j \geqslant 0$. Each request r_j is released at a nonnegative time $t_j \geqslant 0$, its *release time*. For $t \geqslant 0$ we denote by $\sigma_{\leqslant t}$ ($\sigma_{=t}$) the set of requests in σ released no later than time t (exactly at time t). A server is located at the origin $o \in X$ at time 0 and can move at most at unit speed. We consider the case in which the server can serve only one ride at a time, and in which preemption of rides is not allowed.

An online algorithm learns of the existence of request r_j only at its release time t_j. In particular, it has neither information about the release time of the last request nor about the total number of requests. Hence, at any moment in time t, an online algorithm can base its decisions only on the requests in $\sigma_{\leqslant t}$. An offline algorithm has complete knowledge about the sequence σ already at time 0.

Given a sequence σ of requests, a *feasible route* for σ is a sequence of moves of the server such that the following conditions are satisfied: (a) The server starts in the origin o, (b) each ride requested in σ is served, and (c) the repairman does not start to serve any ride (in its source) before its release time. Let C_j^S denote the completion time of request r_j on a feasible route S. The *length* of a route S, denoted by $l(S)$, is defined as the difference between the time when S is completed and its start time.

Definition 1 (L-OLDARP, OLTRP). Given a request sequence $\sigma = r_1, \ldots, r_m$ the problem L-OLDARP is to find a feasible route S minimizing $\sum_{j=1}^{m} w_j C_j^S$. The *online traveling repairman problem* (OLTRP) is the special case of L-OLDARP in which for each request r_j source and destination coincide, i.e., $a_j = b_j$.

We denote by $\mathsf{ALG}(\sigma)$ the objective function value of the solution produced by an algorithm ALG on input σ. We use OPT to denote an optimal offline algorithm.

Definition 2 (Competitive deterministic algorithm). A deterministic online algorithm ALG for the *L-OLDARP* is c-competitive, if there exists a constant c such that for any request sequence σ: $\mathsf{ALG}(\sigma) \leqslant c \cdot \mathsf{OPT}(\sigma)$.

A randomized online algorithm is a probability distribution over a set of deterministic online algorithms. The objective value produced by a randomized algorithm is therefore a random variable. In this paper we analyze the performance of randomized online algorithms against an *oblivious adversary*. An oblivious adversary knows the online algorithm and the distributions it uses, but does not see the realizations of the random choices made by the online algorithm and therefore has to generate a request sequence in advance. We refer to [5] for details on the various adversary models.

Definition 3 (Competitive randomized algorithm). A randomized online algorithm RALG is c-competitive against an oblivious adversary if for any request sequence σ: $\mathbb{E}[\mathsf{RALG}(\sigma)] \leqslant c \cdot \mathsf{OPT}(\sigma)$.

3. A deterministic algorithm

Our deterministic strategy is an adaption of the GREEDY-INTERVAL algorithm presented in [9,10] for online scheduling. The proof of performance borrows concepts of the proofs in [9,10].

Algorithm INTERVAL$_\alpha$

Phase 0: In this phase the algorithm is initialized.

Set L to be the earliest time when a request could be completed by OPT. We can assume that $L > 0$, since $L = 0$ means that there are requests released at time 0 with source and destination o. These requests are served at no cost. For $i = 0, 1, 2, \ldots$, define $B_i := \alpha^{i-1} L$, where $\alpha \in [1 + \sqrt{2}, 3]$ is fixed.

Phase i, for $i = 1, 2, \ldots$: At time B_i compute a transportation schedule S_i for the set of yet unserved requests released up to time B_i with the following properties:

(i) Schedule S_i starts at the endpoint x_{i-1} of schedule S_{i-1} (we set $x_0 := o$).

(ii) Schedule S_i ends at a point x_i with an empty server such that $d(o, x_i) \leqslant B_i$.

(iii) The length of schedule S_i, denoted by $l(S_i)$, satisfies

$$l(S_i) \leqslant \begin{cases} B_1 & \text{if } i = 1 \\ B_i + B_{i-1} & \text{if } i \geqslant 2. \end{cases}$$

(iv) The transportation schedule S_i maximizes the sum of the weights of requests served among all schedules satisfying (i)–(iii).

If $i = 1$, then follow S_1 starting at time B_1. If $i \geqslant 2$, follow S_i starting at time βB_i until βB_{i+1}, where $\beta := (\alpha + 1)/(\alpha(\alpha - 1))$.

To justify the correctness of the algorithm first notice that $\beta \geqslant 1$ for any $\alpha \geqslant 1 + \sqrt{2}$. Moreover, it holds that the transportation schedule S_i computed at time B_i can actually be finished before time βB_{i+1}, the time when transportation schedule S_{i+1}, computed at time B_{i+1}, needs to be started: S_1 is finished latest at time $B_1 + B_1 = 2B_1 \leqslant \frac{\alpha+1}{\alpha-1} B_1 = \beta B_2$ since $\alpha \leqslant 3$. For $i \geqslant 2$, schedule S_i is also finished in time: By condition (iii), $l(S_i) \leqslant B_i + B_{i-1} = (1 + 1/\alpha) B_i$. Hence, schedule S_i is finished latest at time $\beta B_i + (1 + \frac{1}{\alpha}) B_i = (\alpha + 1)/(\alpha - 1) B_i = \beta B_{i+1}$.

Lemma 4. *Let R_i be the set of requests served by schedule S_i computed at time B_i, $i = 1, 2, \ldots$, and let R_i^* be the set of requests in the optimal offline solution which are*

completed in the time interval $(B_{i-1}, B_i]$. *Then*

$$\sum_{i=1}^{k} w(R_i) \geqslant \sum_{i=1}^{k} w(R_i^*) \quad \text{for } k = 1, 2, \ldots .$$

Proof. We first argue that for any $k \geqslant 1$ we can obtain from the optimal offline solution S^* a schedule S which starts in the origin, has length at most B_k, ends with an empty server at a point with distance at most B_k from the origin, and which serves all requests in $\bigcup_{i=1}^{k} R_i^*$.

Consider the optimal offline transportation schedule S^*. Start at the origin and follow S^* for the first B_k time units with the modification that, if a request is picked up in S^* before time B_k but not delivered before time B_k, omit this action. Observe that this implies that the server is empty at the end of this schedule. We thereby obtain a schedule S of length at most B_k which serves all requests in $\bigcup_{i=1}^{k} R_i^*$. Since the server moves at unit speed, it follows that S ends at a point with distance at most B_k from the origin.

We now consider phase k and show that by the end of phase k, at least requests of weight $\sum_{i=1}^{k} w(R_i^*)$ have been scheduled by $\mathsf{INTERVAL}_\alpha$. If $k = 1$, the transportation schedule S obtained as outlined above satisfies already all conditions (i)–(iii) required by $\mathsf{INTERVAL}_\alpha$. If $k \geqslant 2$, then condition (i) might be violated, since S starts in the origin. However, we can obtain a new schedule S' from S starting at the endpoint x_{k-1} of the schedule from the previous phase, moving the empty server from x_{k-1} to the origin and then following S. Since $d(x_{k-1}, o) \leqslant B_{k-1}$, the new schedule S' has length at most $B_{k-1} + l(S) \leqslant B_{k-1} + B_k$ which means that it satisfies all the properties (i)–(iii) required by $\mathsf{INTERVAL}_\alpha$.

Recall that schedule S and thus also S' serves all requests in $\bigcup_{i=1}^{k} R_i^*$. Possibly, some of the requests from $\bigcup_{i=1}^{k} R_i^*$ have already been served by $\mathsf{INTERVAL}_\alpha$ in previous phases. As omitting requests can never increase the length of a transportation schedule, in phase k, $\mathsf{INTERVAL}_\alpha$ can schedule at least all requests from

$$\left(\bigcup_{i=1}^{k} R_i^* \right) \setminus \left(\bigcup_{i=1}^{k-1} R_i \right).$$

Consequently, the weight of all requests served in schedules $S_1, \ldots, S_k$ of $\mathsf{INTERVAL}_\alpha$ is at least $w(\bigcup_{i=1}^{k} R_k^*) = \sum_{i=1}^{k} w(R_k^*)$ as claimed. □

The previous lemma gives us the following bound on the number of phases that $\mathsf{INTERVAL}_\alpha$ uses to process a given input sequence σ.

Corollary 5. *Suppose that the optimum offline schedule is completed in the interval* $(B_{p-1}, B_p]$ *for some* $p \geqslant 1$. *Then the number of phases of the Algorithm* $\mathsf{INTERVAL}_\alpha$ *is at most* p. *Schedule* S_p *computed at time* B_p *by* $\mathsf{INTERVAL}_\alpha$ *is completed no later than time* βB_{p+1}.

Proof. By Lemma 4 the weight of all requests scheduled in the first p phases equals the total weight of all requests. Hence all requests must be scheduled within the first

p phases. Since, by construction of $\mathsf{INTERVAL}_\alpha$, schedule S_p computed in phase p completes by time βB_{p+1}, the claim follows. □

To prove competitiveness of $\mathsf{INTERVAL}_\alpha$ we need an elementary lemma which can be proven by induction.

Lemma 6. *Let $a_i, b_i \in \mathbb{R}_0^+$ for $i = 1, \ldots, p$, for which*
(i) $\sum_{i=1}^{p} a_i = \sum_{i=1}^{p} b_i$;
(ii) $\sum_{i=1}^{p'} a_i \geqslant \sum_{i=1}^{p'} b_i$ *for all* $1 \leqslant p' \leqslant p$.

Then $\sum_{i=1}^{p} \tau_i a_i \leqslant \sum_{i=1}^{p} \tau_i b_i$ for any nondecreasing sequence $0 \leqslant \tau_1 \leqslant \cdots \leqslant \tau_p$.

Theorem 7. *Algorithm $\mathsf{INTERVAL}_\alpha$ is $\alpha(\alpha+1)/(\alpha-1)$-competitive for the* L-OLDARP *for any $\alpha \in [1+\sqrt{2}, 3]$. For $\alpha = 1+\sqrt{2}$, this yields a competitive ratio of $(1+\sqrt{2})^2 < 5.8285$.*

Proof. Let $\sigma = r_1, \ldots, r_m$ be any sequence of requests. By definition of $\mathsf{INTERVAL}_\alpha$, each request served in schedule S_i completes no later than time $\beta B_{i+1} = \alpha + 1/\alpha(\alpha-1)B_{i+1}$. Summing over all phases $1, \ldots, p$ yields

$$\mathsf{INTERVAL}_\alpha(\sigma) \leqslant \frac{\alpha+1}{\alpha(\alpha-1)} \sum_{i=1}^{p} B_{i+1} w(R_i) = \alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{i=1}^{p} B_{i-1} w(R_i). \tag{1}$$

From Lemma 4 we know that $\sum_{i=1}^{k} w(R_i) \geqslant \sum_{i=1}^{k} w(R_i^*)$ for $k = 1, 2, \ldots$, and from Corollary 5 we know that $\sum_{i=1}^{p} w(R_i) = \sum_{i=1}^{p} w(R_i^*)$. Therefore, application of Lemma 6 to the sequences $a_i := w(R_i)$ and $b_i := w(R_i^*)$ with the weighing sequence $\tau_i := B_{i-1}$, $i = 1, \ldots, p$, gives

$$\alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{i=1}^{p} B_{i-1} w(R_i) \leqslant \alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{i=1}^{p} B_{i-1} w(R_i^*). \tag{2}$$

Denote by C_j^* the completion time of request r_j in the optimal offline solution $\mathsf{OPT}(\sigma)$. For each request r_j denote by $(B_{\phi_j}, B_{\phi_{j+1}}]$ the interval that contains C_j^*. Then

$$\alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{i=1}^{p} B_{i-1} w(R_i^*) = \alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{j=1}^{m} B_{\phi_j} w_j \leqslant \alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{j=1}^{m} w_j C_j^*. \tag{3}$$

Eqs. (1)–(3) together yield

$$\mathsf{INTERVAL}_\alpha(\sigma) \leqslant \alpha \cdot \frac{\alpha+1}{\alpha-1} \cdot \mathsf{OPT}(\sigma).$$

The value $\alpha = 1+\sqrt{2}$ minimizes the function $f(\alpha) := \alpha(\alpha+1)/(\alpha-1)$ in the interval $[1+\sqrt{2}, 3]$, yielding $(1+\sqrt{2})^2 < 5.8285$ as competitive ratio. □

Corollary 8. *For $\alpha = 1+\sqrt{2}$, algorithm $\mathsf{INTERVAL}_\alpha$ is $(1+\sqrt{2})^2$-competitive for the* OLTRP.

4. An improved randomized algorithm

In this section we use techniques developed in [6] to devise a randomized algorithm RANDINTERVAL$_\alpha$. At the beginning, RANDINTERVAL$_\alpha$ chooses a random number $\delta \in (0,1]$ according to the uniform distribution. From this moment on, the algorithm is completely deterministic, working in the same way as the deterministic algorithm INTERVAL$_\alpha$ presented in the previous .section For $i \geqslant 0$ define $B'_i := \alpha^{i-1+\delta}L$, where again L is the earliest time that a request could be completed by OPT. As stated before in the case of INTERVAL$_\alpha$ we can assume that $L > 0$.

The difference between RANDINTERVAL$_\alpha$ and INTERVAL$_\alpha$ is that all phases are defined using $B'_i := \alpha^{i-1+\delta}L$ instead of $B_i := \alpha^{i-1}L$, $i \geqslant 1$. To justify the accuracy of RANDINTERVAL$_\alpha$, note that in the correctness proof for the deterministic version, we only made use of the fact that $B_{i+1} = \alpha B_i$ for $i \geqslant 0$. This also holds for the B'_i. Hence, any choice of the parameter $\alpha \in [1+\sqrt{2}, 3]$ yields a correct version of RANDINTERVAL$_\alpha$. We will show later that the optimal choice for RANDINTERVAL$_\alpha$ is $\alpha = 3$.

The proof of Lemma 4 also holds also with B_i replaced by B'_i for each $i \geqslant 0$. We thus obtain the following lemma.

Lemma 9. *Let R_i be the set of requests scheduled in phase $i \geqslant 1$ of Algorithm* RAND INTERVAL$_\alpha$ *and denote by R_i^* the set of requests that are completed by* OPT *in the time interval $(B'_{i-1}, B'_i]$. Then*

$$\sum_{i=1}^{k} w(R_i) \geqslant \sum_{i=1}^{k} w(R_i^*) \quad \text{for } k = 1, 2, \ldots .$$

We can now use the proof of Theorem 7 with Lemma 4 replaced by Lemma 9. This enables us to conclude that for a sequence $\sigma = r_1, \ldots, r_m$ of requests the expected objective function value of RANDINTERVAL$_\alpha$ satisfies

$$\begin{aligned} \mathbb{E}[\text{RANDINTERVAL}_\alpha[(\sigma)] &\leqslant \mathbb{E}\left[\alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{j=1}^{m} B'_{\phi_j} w_j\right] \\ &= \alpha \cdot \frac{\alpha+1}{\alpha-1} \sum_{j=1}^{m} w_j \mathbb{E}[B'_{\phi_j}], \end{aligned} \tag{4}$$

where $(B'_{\phi_j}, B'_{\phi_{j+1}}]$ is the interval containing the completion time C_j^* of request r_j in the optimal solution OPT(σ).

To prove a bound on the performance of RANDINTERVAL$_\alpha$ we compute $\mathbb{E}[B'_{\phi_j}]$. Notice that B'_{ϕ_j} is the largest value $\alpha^{k+\delta}L$, $k \in \mathbb{Z}$, which is strictly smaller than C_j^*.

Lemma 10. *Let $z \geqslant L$ and $\delta \in (0,1]$ be a random variable uniformly distributed on $(0,1]$. Define B by $B := \max\{\alpha^{k+\delta}L : \alpha^{k+\delta}L < z \text{ and } k \in \mathbb{Z}\}$. Then, $\mathbb{E}[B] \leqslant (\alpha - 1)/(\alpha \ln \alpha) \cdot z$.*

Proof. Suppose that $\alpha^k L \leqslant z < \alpha^{k+1} L$ for some $k \geqslant 0$. Observe that

$$B = \begin{cases} \alpha^{k-1+\delta} L & \text{if } \delta \geqslant \log_\alpha \frac{z}{\alpha^k L}, \\ \alpha^{k+\delta} L & \text{otherwise.} \end{cases}$$

Hence

$$\mathbb{E}[B] = \int_0^{\log_\alpha(z/\alpha^k L)} \alpha^{k+\delta} L \, \mathrm{d}\delta + \int_{\log_\alpha(z/\alpha^k L)}^1 \alpha^{k-1+\delta} L \, \mathrm{d}\delta$$

$$= \alpha^k L \left[\frac{1}{\ln \alpha} \alpha^\delta \right]_0^{\log_\alpha(z/\alpha^k L)} + \alpha^{k-1} L \left[\frac{1}{\ln \alpha} \alpha^\delta \right]_{\log_\alpha(z/\alpha^k L)}^1 = \frac{(\alpha - 1)}{(\alpha \ln \alpha)} \cdot z.$$

This completes the proof. □

From Lemma 10 we can conclude that $\mathbb{E}[B'_{\phi_j}] = (\alpha - 1)/(\alpha \ln \alpha) \cdot C_j^*$. Using this result in inequality (4), we obtain

$$\mathbb{E}[\mathsf{RANDINTERVAL}_\alpha(\sigma)] \leqslant \alpha \cdot \frac{\alpha + 1}{\alpha - 1} \frac{\alpha - 1}{\alpha \ln \alpha} \cdot \sum_{j=1}^m w_j C_j^* = \frac{\alpha + 1}{\ln \alpha} \cdot \mathsf{OPT}(\sigma).$$

Minimizing the function $g(\alpha) := (\alpha+1)/(\ln \alpha)$ over the interval $[1+\sqrt{2}, 3]$, we conclude that the best choice is $\alpha = 3$.

Theorem 11. *Algorithm* $\mathsf{RANDINTERVAL}_\alpha$ *is* $(\alpha + 1)/(\ln \alpha)$*-competitive for the* L-OLDARP *against an oblivious adversary, where* $\alpha \in [1+\sqrt{2}, 3]$. *Choosing* $\alpha = 3$ *yields a competitive ratio of* $4/\ln 3 < 3.6410$ *for* $\mathsf{RANDINTERVAL}_\alpha$ *against an oblivious adversary.*

Corollary 12. *For* $\alpha = 3$, *algorithm* $\mathsf{RANDINTERVAL}_\alpha$ *is* $(4/\ln 3)$*-competitive for the* OLTRP *against an oblivious adversary.*

5. Lower bounds

In this section we show lower bounds for the competitive ratio of any randomized algorithm against an oblivious adversary for the problem L-OLDARP. The basic method for deriving such a lower bound is Yao's principle (see also [5,11,12]). Let X be a probability distribution over input sequences $\Sigma = \{\sigma_x : x \in \mathcal{X}\}$. We denote the expected cost of the deterministic algorithm ALG according to the distribution X on Σ by $\mathbb{E}_X[\mathsf{ALG}[(\sigma_x)]$. Yao's principle can now be stated as follows.

Theorem 13 (Yao's principle). *Let* $\{\mathsf{ALG}_y : y \in \mathcal{Y}\}$ *denote the set of deterministic online algorithms for an online minimization problem. If* X *is a distribution over input sequences* $\{\sigma_x : x \in \mathcal{X}\}$ *such that*

$$\inf_{y \in \mathcal{Y}} \mathbb{E}_X[\mathsf{ALG}_y(\sigma_x)] \geqslant \bar{c} \mathbb{E}_X[\mathsf{OPT}(\sigma_x)]$$

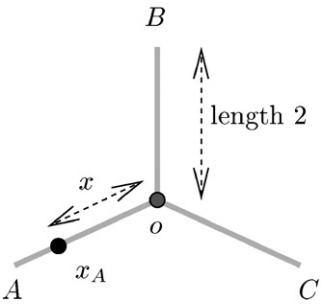

Fig. 1. The star with three rays used in the lower bound construction.

for some real number $\bar{c} \geqslant 1$, then $\bar{c}$ is a lower bound on the competitive ratio of any randomized algorithm against an oblivious adversary.

We use a method explained in [12] to compute a suitable distribution once our ground set of request sequences has been fixed.

5.1. A General lower bound for the L-OLDARP

We provide a general lower bound where the metric space is a star, which consists of three rays of length 2 each. The center of the star is the origin, denoted by o. The server capacity equals one.

Let the rays of the star be named A, B and C. Let x_A be the point on A with distance x to o, $0 < x \leqslant 2$. Points on B and C will be denoted in the same manner. Fig. 1 contains an illustration of the star-shaped metric space.

Let $k \in \mathbb{N}$, $k \geqslant 2$ be arbitrary. At time 0 there is one request from o to 1_A with weight 1, denoted by $r_1 = (0, o, 1_A, 1)$. With probability θ there are no further requests. With probability $1 - \theta$ there are k requests at time $2x$, where $x \in (0, 1]$ is chosen according to a density function p. The density p satisfies $\theta + \int_0^1 p(x)\,\mathrm{d}x = 1$ and will be determined suitably in the sequel. Each of the k requests released at time $2x$ has weight 1. With probability $\frac{1}{2}(1 - \theta)$ these requests will be given from $2x_B$ to $2x_B$ and with probability $\frac{1}{2}(1 - \theta)$ from $2x_C$ to $2x_C$. This yields a probability distribution X over the set $\Sigma = \{\sigma_{x,R}\colon\ x \in [0, 1],\ R \in \{B, C\}\}$ of request sequences where $\sigma_{0,R} = r_1$ for $R \in \{B, C\}$, and

$$\sigma_{x,R} = r_1, \underbrace{(2x, 2x_R, 2x_R, 1), \ldots, (2x, 2x_R, 2x_R, 1)}_{k \text{ times}} \quad \text{for } 0 < x \leqslant 1 \text{ and } R \in \{B, C\}.$$

Fig. 2 illustrates the construction.

We first calculate the expected cost $\mathbb{E}_X[\mathsf{OPT}(\sigma_{x,R})]$ of an optimal offline algorithm with respect to the distribution X on Σ. With probability θ there is only request r_1 to be served, and in this case the offline cost is 1. Now consider the situation where there are k additional requests at position $2x_B$ or $2x_C$. Clearly, the optimal offline cost is

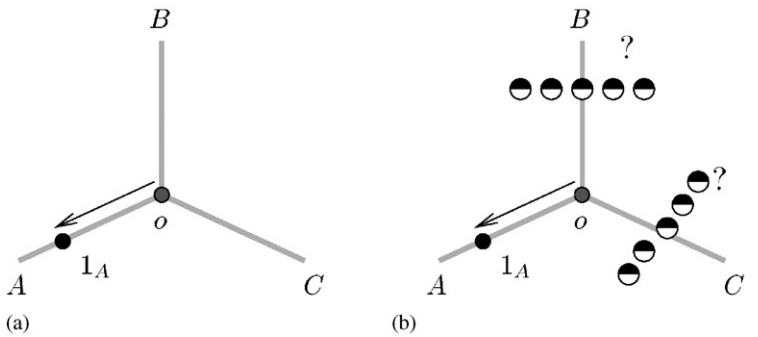

Fig. 2. Lower bound construction: (a) Situation at time 0; (b) a set of k new requests arrives at time $2x$ either in $2x_B$ or $2x_C$, where x is chosen according to a density p such that $\theta + \int_0^1 p(x)dx = 1$.

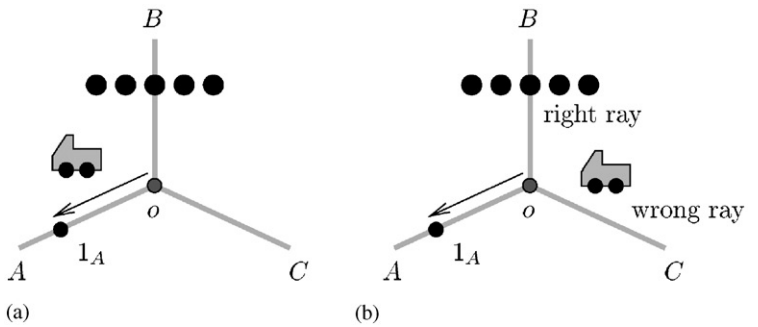

Fig. 3. Two main cases in the lower bound construction: (a) case 1: $2x > 2y$, i.e., new requests arrive after ALG_y has started to serve r_1; (b) case 2: $2x \leqslant 2y$, i.e., new request arrive before ALG_y starts serving r_1.

independent of the ray on which the requests arrive. First serving request r_1 and then the k requests yields the objective function value $1 + k(2 + 2x)$, whereas first serving the set of k requests and then r_1 results in a total cost of $2kx + 4x + 1$. Hence, for $k \geqslant 2$, we have

$$\mathbb{E}_X[\mathsf{OPT}(\sigma_{x,R})] = \theta + \int_0^1 (2kx + 4x + 1)\, p(x)\, \mathrm{d}x. \tag{5}$$

The strategy of a generic deterministic online algorithm ALG_y can be cast into the following framework: ALG_y starts serving request r_1 at time $2y$ where $y \geqslant 0$, unless further requests are released before time $2y$. If the sequence ends after r_1, the online costs are $2y + 1$. Otherwise, two cases have to be distinguished.

If $2x > 2y$, that is, the set of k requests is released after the time at which ALG_y starts the ride requested in r_1, the algorithm must first finish r_1 before it can serve the k requests. In this case, the cost of ALG_y is at least $2y + 1 + k(2y + 2 + 2x)$ (see Fig. 3(a)).

If $2x \leqslant 2y$, the server of ALG_y has not yet started r_1 and can serve the k requests before r_1. To calculate the cost it incurs in this case, let l denote the distance of ALG_y's server to the origin at time $2x$. Then $0 \leqslant l \leqslant y$ since otherwise, the server cannot start r_1 at time $2y$. We may assume that ALG_y is either on ray B or C, since moving onto ray A without serving r_1 is clearly not advantageous.

Now, with probability $\frac{1}{2}$, ALG_y's server is on the "wrong" ray (see Fig. 3(b)). This yields cost of at least $(2x+(l+2x))k+6x+l+1$ for ALG_y. Being on the "right" ray will cost $(2x+(2x-l))k+6x-l+1$. Putting this together, we get that for $y \leqslant 1$:

$$\begin{aligned}\mathbb{E}_X[\mathsf{ALG}_y(\sigma_{x,R})] &\geqslant \theta(2y+1)+\int_y^1 (2y+1+k(2y+2+2x))\,p(x)\,\mathrm{d}x\\ &\quad+\frac{1}{2}\int_0^y (4kx+6x+kl+l+1)\,p(x)\,\mathrm{d}x\\ &\quad+\frac{1}{2}\int_0^y (4kx+6x-kl-l+1)\,p(x)\,\mathrm{d}x.\end{aligned}$$

This results in

$$\begin{aligned}\mathbb{E}_X[\mathsf{ALG}_y(\sigma_{x,R})] &\geqslant \theta(2y+1)+\int_y^1 (2y+1+k(2y+2+2x))\,p(x)\,\mathrm{d}x\\ &\quad+\int_0^y (4kx+6x+1)\,p(x)\,\mathrm{d}x.\\ &=: F(y).\end{aligned} \tag{6}$$

Observe that for $y \geqslant 1$ we have that

$$\mathbb{E}_X[\mathsf{ALG}_y(\sigma_{x,R})] \geqslant \theta(2y+1)+\int_0^1 (4kx+6x+1)\,p(x)\,\mathrm{d}x \geqslant F(1).$$

Hence in what follows it suffices to consider the case $y \leqslant 1$. To maximize the expected cost of any deterministic online algorithm on our randomized input sequence, we wish to choose θ and a density function p such that $\min_{y\in[0,1]} F(y)$ is maximized. We use the following heuristic approach (cf. [12]): Assume that θ and the density function p maximizing the minimum have the property that $F(y)$ is constant on $[0,1]$. Hence $F'(y)=0$ and $F''(y)=0$ for all $y \in (0,1)$. Differentiating we find that

$$F'(y) = 2\left(\theta+(1+k)\int_y^1 p(x)\,\mathrm{d}x-(k-2y)p(y)\right)$$

and

$$F''(y) = -2(k-1)p(y)-2(k-2y)p'(y).$$

From the condition $F''(y)=0$ for all $y \in (0,1)$ we obtain the differential equation

$$-2(k-1)p(x)-2(k-2x)p'(x)=0,$$

which has the general solution

$$p(x) = \beta(k - 2x)^{1/2(k-1)}. \tag{7}$$

The value of $\beta > 0$ is obtained from the initial condition $\theta + \int_0^1 p(x)\,\mathrm{d}x = 1$ as

$$\beta = \frac{1-\theta}{\int_0^1 (k-2x)^{1/2(k-1)}\,\mathrm{d}x} = \frac{(1+k)(\theta-1)}{(k-2)^{(1+k)/2} - k^{(1+k)/2}}. \tag{8}$$

It remains to determine θ. Recall that we attempted to choose θ and p in such a way that F is constant over the interval $[0,1]$. Hence in particular we must have $F(0) = F(1)$. Using

$$F(0) = \theta + \int_0^1 (1 + k(2+2x))\,p(x)\,\mathrm{d}x$$

and

$$F(1) = 3\theta + \int_0^1 (4kx + 6x + 1)\,p(x)\,\mathrm{d}x$$

and substituting p and β from (7) and (8), respectively, we obtain

$$\theta = \frac{(k-2)^{(1+k)/2}(1+k)}{(k-2)^{(1+k)/2}k + k^{(1+k)/2}}.$$

We now use the distribution obtained this way in (6) and (5). This results in

$$\mathbb{E}_X[\mathsf{ALG}_y(\sigma_{x,R})] \geqslant \frac{(1+k)(-5(k-2)^{(2+k)/2}\sqrt{k} + \sqrt{k-2}k^{k/2}(3+4k))}{\sqrt{k-2}(3+k)((k-2)^{(1+k)/2}\sqrt{k} + k^{k/2})}$$

and

$$\mathbb{E}_X[\mathsf{OPT}(\sigma_{x,R})] = \frac{\sqrt{k-2}k^{(1+k)/2}(1+k)(3+2k) - (k-2)^{(2+k)/2}(1+k)(4+3k)}{\sqrt{k-2}((k-2)^{(1+k)/2}k(3+k) + k^{(1+k)/2}(3+k))}.$$

Hence we conclude that

$$\frac{\mathbb{E}_X[\mathsf{ALG}_y(\sigma_{x,R})]}{\mathbb{E}_X[\mathsf{OPT}(\sigma_{x,R})]} \geqslant \frac{-5(k-2)^{1+k/2}k + \sqrt{k-2}k^{(1+k)/2}(3+4k)}{\sqrt{k-2}k^{(1+k)/2}(3+2k) - (k-2)^{(2+k)/2}(4+3k)}. \tag{9}$$

For $k \to \infty$, the right hand side of (9) converges to $(4e-5)/(2e-3)$. Hence by Yao's Principle we obtain the following result:

Theorem 14. *Any randomized algorithm for the* L-OLDARP has a competitive ratio greater or equal to $(4e-5)/(2e-3) > 2.4104$ *against an oblivious adversary*.

5.2. A lower bound on the real line

The lower bound construction on the star uses the fact that the online server does not know on which ray to move if it wants to anticipate the arrival of the k requests at time $2x$. If the metric space is the real line, there are only two rays, and this argument is no longer valid. The server can move towards the point $2x$ (of course still at the risk that there will be no requests at all on this ray) in anticipation of the set of k requests. Essentially the same construction therefore leads to a slightly worse lower bound.

Theorem 15. *Any randomized algorithm for the* L-OLDARP *on the real line has competitive ratio greater or equal to* $(\ln 16+1)/(\ln 16-1)>2.1282$ *against an oblivious adversary*.

Proof. We use the following request set: At time 0 there is one request $r_1=(0,0,-1,1)$. With probability θ there are no further requests.

With probability $1-\theta$ there are k requests at time $2x$ from $2x$ to $2x$ where $x\in(0,1]$ is chosen according to the density function $p(x)$, where $\theta+\int_0^1 p(x)\,\mathrm{d}x=1$. Again, this yields a probability distribution X over a set $\Sigma=\{\sigma_x\colon\ x\in[0,1]\}$ of request sequences where

$$\sigma_x=\begin{cases} r_1 & \text{for } x=0\\ r_1,\underbrace{(2x,2x,2x,1),\ldots,(2x,2x,2x,1)}_{k \text{ times}} & \text{for } 0<x\leqslant 1.\end{cases}$$

By a similar argumentation as in the previous section we obtain

$$\mathbb{E}_X[\mathsf{OPT}(\sigma_x)]=\theta+\int_0^1(2kx+4x+1)\,p(x)\,\mathrm{d}x$$

and

$$\mathbb{E}_X[\mathsf{ALG}_y(\sigma_x)]\geqslant\theta(2y+1)+\int_0^y(2kx+4x+1)\,p(x)\,\mathrm{d}x + \int_y^1(2y+1+k(2y+2+2x))\,p(x)\,\mathrm{d}x.$$

By choosing

$$p(x)=\beta(k-x+kx)^{-2k/(-1+k)},$$

$$\beta=\frac{1-\theta}{\int_0^1(k-x+kx)^{-2k/(-1+k)}\,\mathrm{d}x}=\frac{(1+k)(1-\theta)}{k^{(1+k)/(1-k)}-(-1+2k)^{(1+k)/(1-k)}},$$

and

$$\theta=\frac{-1+k+2k^2}{k(-1+2k+k^{-2k/(-1+k)}(-1+2k)^{2k/(-1+k)})},$$

we obtain

$$\frac{\mathbb{E}_X[\mathsf{ALG}_y(\sigma_x)]}{\mathbb{E}_X[\mathsf{OPT}(\sigma_x)]} \geqslant 1 + \frac{2}{(1+k)(-2+k^{(1+k)/(1-k)}(-1+2k)^{(1+k)/(-1+k)})}. \tag{10}$$

For $k \to \infty$, the right-hand side of (10) converges to $(\ln 16 + 1)/(\ln 16 - 1)$. The theorem now follows from Yao's Principle. □

5.3. Lower bounds for the OLTRP

For the OLTRP we provide a general lower bound, again using a star with three rays of length 2 as a metric space.

Theorem 16. *Any randomized algorithm for the* OLTRP *has competitive ratio greater or equal to* $\frac{7}{3}$ *against an oblivious adversary.*

Proof. Let $k \in \mathbb{N}$ be arbitrary. With probability $\theta = (k+1)/(k+2)$, the input sequence consists of only one request r_1 at distance 1 from the origin, released at time 1, and with weight 1. The ray on which this request occurs is chosen uniformly at random among the three rays.

With probability $1 - \theta$ there will be an additional series of k requests at distance 2 from the origin, each with weight equal to 1. These requests are released at time 2, and the ray on which they occur is chosen uniformly among the two rays that do not contain r_1.

The cost for the adversary is given by

$$\mathbb{E}_X[\mathsf{OPT}(\sigma_x)] = \theta(1-\theta)(2k+5) = \frac{3k+6}{k+2}.$$

It is easy to show that no online algorithm can do better than one whose server is in the origin at time 1 and, at time 2, at distance $0 \leqslant y \leqslant 1$ from the origin on the ray where r_1 is located. Using this fact, we get

$$\mathbb{E}_X[\mathsf{ALG}_y(\sigma_x)] \geqslant \theta(3-y) + (1-\theta)((4+y)k + 7 + y) = \frac{7k+10}{k+2}.$$

This leads to

$$\frac{\mathbb{E}_X[\mathsf{ALG}_y(\sigma_x)]}{\mathbb{E}_X[\mathsf{OPT}(\sigma_x)]} \geqslant \frac{7k+10}{3k+6}. \tag{11}$$

By letting $k \to \infty$ and applying Yao's Principle once more, the theorem follows. □

On the real line, a lower bound is obtained by the following very simple randomized request sequence. With probability $\frac{1}{2}$ we give a request at time 1 in -1, and with probability $\frac{1}{2}$ we give a request at time 1 in $+1$. This leads to the following theorem.

Theorem 17. *Any randomized algorithm for the* OLTRP *on the real line has competitive ratio greater or equal to* 2 *against an oblivious adversary.*

Acknowledgements

The authors would like to thank Günther Rote (FU Berlin) and René Sitters (University of Eindhoven) for independently suggesting to choose an optimized base for the exponentially growing intervals in our algorithms. This led to an improvement in the competitive ratios of both the deterministic and randomized algorithm.

References

[1] F. Afrati, C. Cosmadakis, C. Papadimitriou, G. Papageorgiou, N. Papakostantinou, The complexity of the traveling repairman problem, Informat. Theor. Appl. 20 (1) (1986) 79–87.
[2] S. Arora, G. Karakostas, Approximation schemes for minimum latency problems, Proc. 31st Annu. ACM Symp. on Theory of Computing, 1999, pp. 688–693.
[3] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, M. Talamo, Algorithms for the on-line traveling salesman, Algorithmica 29 (4) (2001) 560–581.
[4] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, Proc. 26th Annu. ACM Symp. Theory of Computing, 1994, pp. 163–171.
[5] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, Cambridge, 1998.
[6] S. Chakrabarti, C.A. Phillips, A.S. Schulz, D.B. Shmoys, C. Stein, J. Wein, Improved scheduling algorithms for minsum criteria, Proc. 23rd Internat. Colloq. on Automata, Languages and Programming, Lecture Notes in Computer Science, Vol. 1099, Springer, Berlin, 1996, pp. 646–657.
[7] E. Feuerstein, L. Stougie, On-line single server dial-a-ride problems, Theoret. Comput. Sci. 268 (2001) 91–105.
[8] M. Goemans, J. Kleinberg, An improved approximation ratio for the minimum latency problem, Proc. 7th Annu. ACM-SIAM Symp. on Discrete Algorithms, 1996, pp. 152–158.
[9] L.A. Hall, A.S. Schulz, D.B. Shmoys, J. Wein, Scheduling to minimize average completion time: off-line and on-line approximation algorithms Math. Oper. Res. 22 (1997) 513–544.
[10] L. Hall, D.B. Shmoys, J. Wein, Scheduling to minimize average completion time: off-line and on-line algorithms, Proc. 7th Annu. ACM–SIAM Symp. on Discrete Algorithms, 1996, pp. 142–151.
[11] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[12] S. Seiden, A guessing game and randomized online algorithms, Proc. 32nd Annu. ACM Symp. on the Theory of Computing, 2000, pp. 592–601.