Algorithmica (2004) 40: 319–329
DOI: 10.1007/s00453-004-1116-z

Algorithmica

On-Line Dial-a-Ride Problems Under a Restricted Information Model

Maarten Lipmann,[1] Xiwen Lu,[1,2] Willem E. de Paepe,[3] Rene A. Sitters,[1] and Leen Stougie[1,4]

Abstract. In on-line dial-a-ride problems servers are traveling in some metric space to serve requests for rides which are presented over time. Each ride is characterized by two points in the metric space, a *source*, the starting point of the ride, and a *destination*, the endpoint of the ride. Usually it is assumed that at the release of a request, complete information about the ride is known. We diverge from this by assuming that at the release of a ride, only information about the source is given. At visiting the source, the information about the destination will be made available to the servers. For many practical problems, our model is closer to reality. However, we feel that the lack of information is often a *choice*, rather than inherent to the problem: additional information *can* be obtained, but this requires investments in information systems. In this paper we give mathematical evidence that for the problem under study it pays to invest.

Key Words. On-line optimization, Competitive analysis, Dial-a-ride.

1. Prelude. In dial-a-ride problems servers are traveling in some metric space to serve requests for rides. Each ride is characterized by two points in the metric space, a *source*, the starting point of the ride, and a *destination*, the endpoint of the ride. The problem is to design routes for the servers through the metric space, such that all requested rides are made and some optimality criterion is met.

Dial-a-ride problems have been studied extensively in the literature of operations research, management science, and combinatorial optimization. Traditionally, such combinatorial optimization problems are studied under the assumption that the input of the problem is known completely to the optimizer.

In a natural setting of dial-a-ride problems requests for rides are presented over time while the servers are enroute serving other rides, making the problem an on-line optimization problem. Examples in practice are taxi and minibus services, courier services, and elevators. In their on-line setting dial-a-ride problems have been studied in [1] and [4], where single-server versions of the problem are studied, as we will do here. These papers study the problem in which rides are specified completely upon presentation, i.e., both source and destination of the ride become known at the same time. We diverge from this setting here.

[1] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. {m.lipmann,x.lu,r.a.sitters,l.stougie}@tue.nl.
[2] East China University of Science and Technology, Shanghai 200237, China. xwlu@ecust.edu.cn.
[3] Department of Technology Management, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. w.e.d.paepe@tm.tue.nl.
[4] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands. stougie@cwi.nl.

Received March 2003; revised December 2003. Communicated by S. Leonardi.
Online publication August 31, 2004.

In many practical situations a complete specification of rides is not realistic. Think for example of the problem to schedule an elevator. Here a ride is the transportation of a person from one floor (the source) to another (the destination), and the release time of the ride is the moment the button on the wall outside the elevator is pressed. The destination of the ride is revealed only when the person enters the elevator and presses the button inside the elevator.

In this paper we study the on-line single-server dial-a-ride problem in which only the source of a ride is presented at the release time of the ride. The destination of a ride is revealed at visiting its source. We call this model the *incomplete ride information model* and refer to the model used in [1] and [4] as the *complete ride information model*. As the objective we minimize the time by which the server has executed all the rides and returned to the origin.

In an M.Sc. thesis Seleson introduced the idea of rides with unknown destinations within a multi-threaded on-line optimization setting [6]. The idea dates back to the mid-nineties and was proposed by her supervisor E. Feuerstein.

We distinguish two versions of the on-line dial-a-ride problem under the incomplete ride information model. In the first version, the *preemptive version*, the server is allowed to preempt any ride at any point, and proceed the ride later. In particular, the server is allowed to visit the source of a ride and learn its destination without executing the ride immediately. In the second version, the *non-preemptive* version, a ride has to be executed as soon as the ride has been picked up in the source. We do allow the server to pass a source without starting the ride, in which case he does not learn the destination of the ride at passing the source. We study each version of the problem under various *capacities* of the server. The capacity of a server is the number of rides the server can execute simultaneously. Problems are defined formally in Section 2.

We perform competitive analysis of deterministic algorithms for the problems described above. Competitive analysis measures the performance quality of an algorithm for an on-line problem by the *competitive ratio*, which is the worst-case ratio over all possible input sequences of the objective value it produces and the optimal off-line solution value. For a detailed explanation of competitive analysis and many examples refer to [3]. For an overview of results on on-line optimization problems refer to [5]. Typically there are lower bounds on the competitive ratio achievable by any algorithm (even allowing exponential computing time). We derive such lower bounds for deterministic algorithms for the various versions of the on-line dial-a-ride problem under the incomplete ride information model. We also design and analyze algorithms for their solution.

In [2] a lower bound of 2 on the competitive ratio of any deterministic algorithm is given for the on-line dial-a-ride problem under the complete ride information model, independent of the capacity of the server, not allowing preemption of rides. However, the bound is based on the on-line traveling salesman problem, having rides with zero length, whence the bound also holds when allowing preemption. In Section 3 we show that under the incomplete ride information model no deterministic preemptive algorithm is better than 3-competitive against an adversary that is not allowed to preempt. This is independent of the capacity of the server. Hence, we have a lower bound of 3 for both the preemptive and non-preemptive version. For the preemptive version, we design an algorithm with a competitive ratio matching the lower bound of 3, independent of the capacity of the server.

Table 1. Overview of lower bounds (LB) and upper bounds (UB) on the competitive ratio of deterministic algorithms for on-line dial-a-ride problems.

	Capacity	LB	UB
Complete ride information			
Preemption	$1, c, \infty$	2 [2]	2 [1]
No preemption	$1, c, \infty$	2 [2]	2 [1]
Incomplete ride information			
Preemption	$1, c, \infty$	3	3
No preemption	1	$1 + \frac{3}{2}\sqrt{2}$	4
	c	$\max\{1 + \frac{3}{2}\sqrt{2}, c\}$	$2c + 2$
	∞	3	3

If preemption is not allowed, we derive a lower bound of $\max\{c, 1 + \frac{3}{2}\sqrt{2}\}$ on the competitive ratio of any deterministic algorithm, where c is a given fixed capacity of the server. We present a $(2c + 2)$-competitive algorithm for the non-preemptive version. These results are presented in Section 4.

We notice that there is no difference between the preemptive version and the non-preemptive version of the problem if the server has infinite capacity, whence we inherit the matching lower and upper bound of 3 of the preemptive version for this case. An overview of the results is given in Table 1. For the exposition we have omitted to refer to [4] for the first 2-competitive algorithm for the problem with complete ride information and server capacity ∞. The result in that paper follows from the results in [1].

Our results combined with those from [1] show the effect of having complete knowledge about rides on worst-case performance for on-line dial-a-ride problems. This is an important issue, since in practice complete information is often lacking. Investments in information systems can help to obtain more information. Mathematical support is essential in justifying such investments. Our results concern minimizing the time by which the server has done all rides and is back in the origin. It is interesting to see if similar results can be obtained for other objectives.

We conclude by referring back to the elevator scheduling problem. The typical elevator with only a request button outside the elevator fits our incomplete ride information model. In an alternative construction, the destination buttons could be built outside the elevator, fitting the complete ride information model. We disclaim though that minimizing the latest completion time is a natural objective for an elevator.

2. Problem Definition. An instance of the on-line single-server dial-a-ride problem (OLDARP) is specified by a metric space $M = (X, d)$ with a distinguished origin $O \in X$, a sequence $\sigma = \sigma_1, \ldots, \sigma_m$ of requests for rides, and a capacity for the server. A server is located at the origin O at time 0 and can move at most at unit speed. We assume that M has the property that for any pair of points $\{x, y\} \in X$ 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 [2] for a thorough discussion of this model). Explicitly, we add the assumption that d is symmetric

and satisfies the triangle inequality for those readers for whom this is not implicit in the definition of metric space.

Each *ride* is a triple $\sigma_i = (t_i, s_i, d_i)$, where $t_i \in \mathbb{R}_0^+$ is the time at which ride σ_i is released, $s_i \in X$ is the source of the ride, and $d_i \in X$ is the destination of the ride. Every ride $\sigma_i \in \sigma$ has to be executed (served); that is, the server has to visit the source, pick up the ride, and end it at the destination. The *capacity* of the server is an upper bound on the number of rides the server can execute simultaneously. We consider unit capacity, constant capacity $c \geq 2$, and infinite capacity for the server. The objective in the OLDARP is to minimize the completion time of the server, which is the time when the server has served all rides and returned to the origin.

We consider the preemptive and non-preemptive versions of the OLDARP, under the *incomplete ride information model* for different capacities of the server.

DEFINITION 2.1. Under the *incomplete ride information model* only the source s_i of ride σ_i is revealed at time t_i. The destination d_i of the ride becomes known only at picking up the ride in the source.

We assume that the sequence $\sigma = \sigma_1, \ldots, \sigma_m$ of rides is given in order of non-decreasing release times, and that the on-line server has neither information about the time when the last ride is released, nor about the total number of rides. An on-line algorithm for the OLDARP must determine the behavior of the server at any moment t based on the information obtained before t, whereas the off-line algorithm knows the whole input sequence σ at time 0. A feasible on-line/off-line solution is a route for the server that starts and ends in the origin O and serves all requested rides regarding that each ride is picked up at the source not earlier than the time it is released.

Let $\mathrm{ALG}(\sigma)$ denote the completion time of the server moved by algorithm ALG on the sequence σ of rides and let $\mathrm{OPT}(\sigma)$ denote the optimal off-line algorithm's completion time. The competitive ratio of algorithm ALG is

$$\max_{\sigma \in \Sigma} \frac{\mathrm{ALG}(\sigma)}{\mathrm{OPT}(\sigma)},$$

with Σ the class of all possible request sequences.

3. The Preemptive Version. We describe our algorithm SNIFFER, which preempts rides only immediately at the source, just to learn the destinations of the rides: it "sniffs" the rides. Upon visiting the source of a ride for the second time, the ride is completed right away. The algorithm is an adaption of the 2-competitive algorithm for the on-line traveling salesman problem (OLTSP) described in [2]. Any time the server is in the origin O it starts an optimal TSP tour over all unvisited sources, just to learn the destinations. If the server is back in O and there are no unvisited sources, then it starts an optimal dial-a-ride (DAR) tour over all rides that still have to be executed. The server ignores all new requests while it is making a TSP or DAR tour. We add the restriction that the server does not start a tour at a time t if the length of this tour is strictly larger than t.

Algorithm SNIFFER

(1) Wait in O until the set S of unvisited sources is non-empty.
(2) Compute an optimal TSP tour $T_{\mathrm{TSP}}(S)$ over the set S of unvisited sources. If the current time is at least $|T_{\mathrm{TSP}}(S)|$ go to (3). Otherwise, wait either until time $|T_{\mathrm{TSP}}(S)|$ and then go to (3) or until a new request is released, then update S and start (2) anew.
(3) Execute $T_{\mathrm{TSP}}(S)$ without interruption. Sources visited are deleted from S and the corresponding rides are added to R, whereas sources of requested rides released during the tour are added to S. At the end of the tour, being back in O, go to (4) if $S = \emptyset$, else go to (2).
(4) Compute an optimal DAR tour $T_{\mathrm{DAR}}(R)$ over the set R of rides that still have to executed. If the current time is at least $|T_{\mathrm{DAR}}(R)|$ go to (5). Otherwise, wait either until time $|T_{\mathrm{DAR}}(R)|$ and then go to (5) or until a new request is released, then update S and go to (2).
(5) Execute $T_{\mathrm{DAR}}(R)$ without interruption, setting $R = \emptyset$. Sources of requested rides released during the tour are added to S. At the end of the tour, being back in O, go to (1) if $S = \emptyset$, else go to (2).

THEOREM 1. SNIFFER *is* 3*-competitive for the preemptive* OLDARP *under the incomplete ride information model, independent of the capacity of the server.*

PROOF. Let S be the set of sources visited in the last TSP tour and let R be the set of rides executed in the last DAR tour. We distinguish three cases.

First, assume that SNIFFER is waiting in O just before it starts the last DAR tour. Then it starts this tour exactly at time $|T_{\mathrm{DAR}}(R)|$, whence the overall completion time is $2|T_{\mathrm{DAR}}(R)| \leq 2|T_{\mathrm{DAR}}(\sigma)|$.

Secondly, assume SNIFFER does not wait before the last DAR tour, but it does wait in O just before the last TSP tour. In this case it starts this tour exactly at time $|T_{\mathrm{TSP}}(S)|$, whence the overall completion time is $|T_{\mathrm{TSP}}(S)| + |T_{\mathrm{TSP}}(S)| + |T_{\mathrm{DAR}}(R)| \leq 3|T_{\mathrm{DAR}}(\sigma)|$.

Finally, assume that SNIFFER waits neither before starting the last TSP tour nor before starting the last DAR tour. Thus, some tour T has been made before the last TSP tour, unless the last TSP tour started at time 0. In the latter case both the TSP and the DAR tour must have length 0. In the former case the tour T might be a TSP or a DAR tour. The set S of requests served on the last TSP tour (and DAR tour) must have been released after the time, t say, that SNIFFER started tour T. Let P be the length of the shortest path through S and O. Then $|T_{\mathrm{DAR}}(\sigma)| \geq t + P$. On the other hand, $|T_{\mathrm{TSP}}(S)| \leq 2P$ implying an overall completion time of

$$t + |T| + |T_{\mathrm{TSP}}(S)| + |T_{\mathrm{DAR}}(R)| \leq 2t + 2P + |T_{\mathrm{DAR}}(R)| \leq 3|T_{\mathrm{DAR}}(\sigma)|. \qquad \square$$

We show that SNIFFER is a best possible deterministic algorithm for the preemptive version of the OLDARP, even though SNIFFER uses preemption only at the source of rides.

THEOREM 2. *No deterministic algorithm can have a competitive ratio strictly smaller than* 3 *for the* OLDARP *under the incomplete ride information model, independent of the capacity of the server.*

PROOF. For the proof of this theorem we use a commonly applied setting of a two-person game, with an adversary providing a sequence of rides, and an on-line algorithm serving the rides (see [3]). Typically, the outcome of the algorithm is compared with the solution value the adversary achieves himself on the sequence, which is in our case the optimal off-line solution value. We consider the OLDARP under the incomplete ride information model where the on-line server has infinite capacity. Let ALG be a deterministic on-line algorithm for this problem. We construct an adversarial sequence σ of requests for rides. We restrict the adversary by giving his server capacity 1. We prove that ALG cannot be strictly better than 3-competitive for this restricted adversary model.

The metric space $M = (X, d)$ contains the set of points, or vertices, $\{x_1, x_2, \ldots, x_{n^2}\} \cup O$ and the distance function d, where $d(O, x_i) = 1$ and $d(x_i, x_j) = 2$ for all x_i, x_j. To facilitate the exposition we denote point x_i by i.

At time 0 there is one ride in each of the points in $1, 2, \ldots, n^2$. If the on-line server visits the source i of a ride at time t with $t \leq 2n^2 - 1$, then the destination turns out to be i as well, and at time $t + 1$ a new ride with source i is released.

In this way the situation remains basically the same for the on-line server until time $2n^2$. We may assume that at some moment t^*, with $2n^2 - 1 < t^* \leq 2n^2$, there is exactly one ride $\sigma_i = (t_i, i, d_i)$ in each point i. Without loss of generality we assume that the points are ordered such that $t_1 \leq \cdots \leq t_{n^2}$.

Thus, at time t^* the on-line server still has to complete exactly n^2 rides. We partition the set of n^2 vertices into n sets: $I_k = \{(k-1)n + 1, \ldots, kn\}$, $k = 1, \ldots, n$. Within each of these sets we order the vertices by the on-line server's first visit to them after time t^*. Let b_{kj}, $j \in \{1, \ldots, n\}$, be the jth vertex in this order in I_k. For all $k \in \{1, \ldots, n\}$ we define $d_{b_{k1}} = b_{k1}$ and $d_{b_{kj}} = b_{k,j-1}$ for all $j \in \{2, \ldots, n\}$. Notice that the destination of ride σ_i only depends on the tour followed by the on-line server until he picks up the ride to look at its destination. For the on-line server this means that n of the n^2 rides can be served immediately since the source equals the destination. For the other $n^2 - n$ rides the server finds out that the destination of the rides he just picked up is another point that he already visited after time t^*. Therefore, $n^2 - n$ points will have to be visited by the on-line server at least twice after time t^*. Hence, the completion time for the on-line server is at least $t^* + 4(n^2 - n) - 1 + 2n > 6n^2 - 2n - 2$.

We now describe the tour made by the adversary. Given our definition of t^* we have that $t_{n^2} \leq t^* \leq 2n^2$. Since the on-line server needs at least 2 time units to move from a point i to another point i', it follows that $t_i \leq 2i$, for all $i \in \{1, \ldots, n^2\}$. The adversary waits until time $2n$ and then starts to serve the rides $\sigma_1, \ldots, \sigma_n$, by visiting the sources in reversed order of $b_{11}, \ldots, b_{1n}$. The rides with equal source and destination are served immediately at arrival in the point. This takes the adversary $2n$ time units. At time $4n$ the adversary starts serving the rides $\sigma_{n+1}, \ldots, \sigma_{2n}$, and then at time $6n$ the rides $\sigma_{2n+1}, \ldots, \sigma_{3n}$, etc. Continuing like this the adversary completes at time $2n^2 + 2n$.

Hence, the competitive ratio is bounded from below by $(6n^2 - 2n - 2)/(2n^2 + 2n)$, which can be made arbitrarily close to 3 by choosing n large enough. □

4. The Non-Preemptive Version. For the non-preemptive version we design an algorithm, called BOUNCER, because the server "bounces" back to the source once a ride is completed. The algorithm uses an algorithm for the OLTSP problem to construct,

on-line, a TSP tour on the sources of the requests. The instance of the OLTSP problem consists only of the sources with their release dates. BOUNCER follows exactly this tour, but being in the source of a ride, it makes the ride and returns to the source, where it proceeds on the TSP tour along all the sources. Since the server in BOUNCER is always behind the OLTSP server this algorithm is well defined. In the analysis we assume that BOUNCER uses a (best possible) 2-competitive algorithm to construct the TSP tour, e.g., the algorithm from [2].

Algorithm BOUNCER

Perform the OLTSP algorithm on the sources of the rides, yielding a tour T along the sources only. Follow T. At each source on T execute the ride, and return to the source via the shortest path. Back at the source proceed on T.

THEOREM 3. BOUNCER *is* $(2c+2)$*-competitive for the* OLDARP *under the incomplete ride information model, where* c *is the capacity of the server.*

PROOF. Consider any request sequence σ. Since all sources have to be visited and the OLTSP algorithm is 2-competitive, $\mathrm{OPT}(\sigma) \geq |T|/2$. Another lower bound is given by $\mathrm{OPT}(\sigma) \geq D/c$, where D is the sum of the lengths of all rides. The completion time of BOUNCER is at most $T + 2D \leq 2\mathrm{OPT}(\sigma) + 2c\mathrm{OPT}(\sigma)$. □

COROLLARY 4.1. BOUNCER *is* 4*-competitive for the* OLDARP *under the incomplete ride information model, if the capacity of the server is* 1.

THEOREM 4. *No non-preemptive deterministic on-line algorithm can have a competitive ratio strictly smaller than* c *for the* OLDARP *under the incomplete ride information model, where* c *is the capacity of the server.*

PROOF. For our lower bound we use the star graph, as we did in the proof of Theorem 2, with $K >> c$ leaves at distance 1 from the origin O. At time 0, cK rides are released, all with their source in O, and each of the leaves being destination of c rides, yielding K sets of c identical rides each. Hence the instance has an optimal solution value $2K$.

Denote the leaves by $1, 2, \ldots, K$. Since the on-line server cannot distinguish between the rides we may assume that the destinations of the rides picked up by the on-line server are consecutively $1, 2, \ldots, K, 1, 2, \ldots$. Consider any of the subsequences $1, \ldots, K$. Before the server picks up the ride with destination K it must have completed at least $K - c$ of the last $K - 1$ rides that it picked up. Therefore the completion time of the on-line server is at least $2c(K - c)$ and the competitive ratio is bounded from below by $2c(K - c)/2K = c - c^2/K$, which can be made arbitrarily close to c by choosing K large enough. □

Together with Theorem 1 this theorem shows that for servers with a capacity greater than 3, the best possible deterministic on-line algorithm for the non-preemptive version of the problem has a strictly higher competitive ratio than SNIFFER for the preemptive

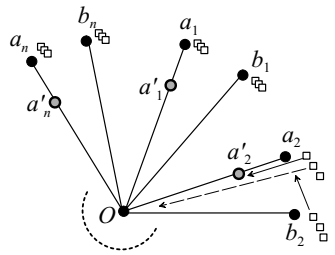

Fig. 1. Lower bound instance. Each leaf contains three sources.

problem. The following theorem shows that this phenomenon also occurs for lower capacities of the server.

THEOREM 5. *No non-preemptive deterministic algorithm can have a competitive ratio strictly smaller than* $1 + \frac{3}{2}\sqrt{2} \approx 3.12$ *for the* OLDARP *under the incomplete ride information model, independent of the capacity of the server.*

PROOF. First we consider the problem when the on-line server has capacity 1. Then we sketch how to extend the proof for any capacity c.

The metric space is defined by the edges of a star graph on $2n + 1$ vertices. The leaves, denoted by a_i $(i = 1, \ldots, n)$ and b_i $(i = 1, \ldots, n)$, have distance 1 to the center (origin) O. On each edge (O, a_i) $(i = 1, \ldots, n)$ we add a point a'_i at a distance α from a_i (Figure 1). The constant α is chosen appropriately later.

The adversary provides the following sequence σ of rides. At time 0 there are three rides in each point a_i and b_i, $i = 1, \ldots, n$. If the on-line server visits a source, then the destination turns out to be the same as the source. Rides of this kind are called *empty* rides. One time unit after an empty ride has been executed the ride is replaced by a new ride with the same source. Every source that is visited by the on-line server strictly before time $4n - 3$ meets the same fate. Sources visited after this time are not replaced. We refer to the three rides that were executed last in a leaf as the *decisive rides* and specify them later.

Since it takes at least two time units to travel between two leaves, the on-line server can visit at most $2n - 2$ leaves during the half-open interval $[0, 4n - 3)$. This leaves two leaves unvisited, which the adversary manipulates to be a_1 and b_1. The adversary will execute the rides in these points first. Similarly, during the interval $[4, 4n - 3)$ only $2n - 4$ leaves can be visited by the on-line server. This leaves two other leaves unvisited after time 4, which the adversary manipulates to be a_2 and b_2. The rides in these points are executed by the adversary after the rides in a_1 and b_1. This receipt is iterated: in each interval $[4(i - 1), 4n - 3)$ there are two leaves that are left unvisited from among those that could have been visited during $[4(i - 2), 4n - 3)$, which the adversary manipulates to be a_i and b_i. Since after time $4(i - 1)$ no more rides are given with sources in a_i and b_i, the adversary executes all rides (including the decisive rides), first the ones related to the pair a_1, b_1, then to the pair a_2, b_2, etc., starting at time 0. The on-line server, however, does not pick up any decisive ride before time $4n - 3$.

We now specify the decisive rides. In each point b_i two of the decisive rides are empty and one has destination a_i. In point a_i one of the decisive rides is empty, one has destination a_i', and one is either empty or has destination O, depending on the actions taken by the on-line server. Without loss of generality we may assume that, from time $4n-3$, the on-line server visits point a_i before point b_i, since before this time he has seen only empty rides and therefore is unable to distinguish between a- and b-leaves. The adversary chooses the rides such that the first ride that the on-line server picks up in point a_i is the ride to a_i'. The first ride the server picks up in point b_i is the ride to a_i. We distinguish between two cases.

Case 1. The on-line server executes the ride from b_i to a_i before it picks up the second ride in a_i. In this case the second ride in a_i is a ride to the origin. The on-line server needs at least 10 time units (from O to O) to serve all the decisive rides connected to the pair a_i, b_i. The adversary serves empty rides at no extra cost and therefore all rides in only $4+2\alpha$ time units (from O to O).

Case 2. The on-line server picks up the second ride in a_i before moving to b_i. In this case the second ride picked up in a_i is empty. The on-line server needs at least $8+2\alpha$ time units to pick up all rides connected to the pair a_i, b_i, whereas the adversary needs only 4.

Starting from O, the on-line server cannot start the decisive rides until time $4n-4$. Let k be the number of pairs a_i, b_i that is served as in case 1. The completion time for the on-line server is at least

$$4n-4+10k+(8+2\alpha)(n-k),$$

and the optimal off-line completion time is

$$(4+2\alpha)k+4(n-k).$$

Standard calculus tells us that, for fixed α, the ratio between these two values is minimized for $k=0$ or for $k=n$. Hence, the competitive ratio is at least

$$\min\left\{\frac{4n-4+10n}{(4+2\alpha)n}, \frac{4n-4+(8+2\alpha)n}{4n}\right\},$$

which tends to

$$\min\left\{\frac{14}{4+2\alpha}, \frac{12+2\alpha}{4}\right\}, \quad \text{as} \quad n\to\infty.$$

For $\alpha = 3\sqrt{2}-4$ this limit is equal to $1+\frac{3}{2}\sqrt{2}$.

If the capacity of the server is c, $c>1$, we give c copies of the same sequence σ simultaneously. An on-line server cannot benefit from this extra capacity in combining rides from different pairs a_i, b_i. The on-line server will have to do the rides in a point in the same order as before. For example, the first c rides that the on-line server picks up in a_i are rides to a_i'. Hence, the completion time for the on-line server cannot be smaller than in the unit capacity case. The off-line server can complete in exactly the same time. □

COROLLARY 4.2. *No non-preemptive deterministic algorithm can have a competitive ratio strictly smaller than* $\max\{1 + \frac{3}{2}\sqrt{2}, c\}$ *for the* OLDARP *under the incomplete ride information model, where c is the capacity of the server.*

5. Postlude. In [1] and [4] the competitive ratio measures the cost of having no information about the release times of future rides. We conclude this paper by discussing below how we can measure the cost of having no information about the destinations of the rides through the competitive ratio.

Suppose that at time 0 the release times and the location of the sources of the rides are given, but the information about the destinations is again revealed only at visiting the sources. Both SNIFFER and BOUNCER use the on-line algorithm of Ausiello et al. [2] for a TSP tour along the sources. In case all sources of the rides and the release times are known, an optimal TSP tour over the sources, that satisfies the release time constraints, can be computed (disregarding complexity issues). In this way SNIFFER and BOUNCER gain an additive factor of 1 on their competitive ratio, making SNIFFER 2-competitive and BOUNCER $(2c + 1)$-competitive.

Notice that the lower bound on the competitive ratio for the non-preemptive problem in Theorem 4 is obtained through a sequence of rides all with release time 0. Thus, this lower bound is completely due to the lack of information about the destinations of the rides.

The rides in the sequence giving the lower bound of $1 + \frac{3}{2}\sqrt{2}$ for the non-preemptive problem in Theorem 5 have release times no larger than $4n - 5$. Taking the unserved rides at time $4n - 5$ as an instance given at time 0 shows that the competitive ratio is at least $\min\{10/(4 + 2\alpha), (8 + 2\alpha)/4\}$. Optimizing over α yields a lower bound of $\frac{1}{2} + \frac{1}{2}\sqrt{11} \approx 2, 15$. Thus, due to the lack of information about destinations only, any algorithm will not be able to attain a ratio of strictly less than $\max\{\frac{1}{2} + \frac{1}{2}\sqrt{11}, c\}$.

In the lower bound construction for the preemptive problem in Theorem 2 the adversary stops giving requests at time $2n^2$. Take the set of rides unserved by any on-line algorithm at that time as an instance with release time 0. Following the proof of Theorem 2 any on-line algorithm will need $4n^2 - 2n$, whereas an optimal tour takes $2n^2$, yielding a lower bound of 2.

Notice that the above lower bounds are established on sequences where all rides have release time 0. For the preemptive version of the problem this is sufficient since the performance of SNIFFER matches the lower bound. However, for the non-preemptive version higher lower bounds might be obtained using diverse release times of rides.

References

[1] N. Ascheuer, S.O. Krumke, and J. Rambau, Online dial-a-ride problems: minimizing the completion time, *Proceedings of the* 17*th International Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, vol. 1770, Springer-Verlag, Berlin, 2000, pp. 639–650.

[2] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo, Algorithms for the on-line traveling salesman, *Algorithmica* **29** (2001), 560–581.

[3] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, Cambridge, 1998.

[4] E. Feuerstein and L. Stougie, On-line single server dial-a-ride problems, *Theoretical Computer Science* **268**(1) (2001), 91–105.
[5] A. Fiat and G.J. Woeginger (eds.), *Online Algorithms*: *The State of the Art*, Lecture Notes in Computer Science, vol. 1442, Springer-Verlag, Berlin, 1998.
[6] M. Seleson, On-line multi-threaded dial-a-ride, M.Sc.- thesis, Facultad de Ciencias y Naturales, Universidad de Buenos Aires, 1997.