Online multi-server dial-a-ride problems

Vincenzo Bonifaci[1,2*], Maarten Lipmann[1], and Leen Stougie[1,3**]

[1] Department of Mathematics and Computer Science
Eindhoven University of Technology
Den Dolech 2 – PO Box 513, 5600 MB Eindhoven, The Netherlands.
`v.bonifaci@tue.nl, m.lipmann@tue.nl, l.stougie@tue.nl`
[2] Department of Computer and Systems Science
University of Rome "La Sapienza"
Via Salaria, 113 – 00198 Roma, Italy.
`bonifaci@dis.uniroma1.it`
[3] CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands.
`stougie@cwi.nl`

Abstract. In an online dial-a-ride problem, a crew of servers has to process transportation requests as they arrive in real time. Possible objective functions include minimizing the makespan and minimizing the sum of completion times. We give competitive algorithms and negative results for several online dial-a-ride problems with multiple servers. Surprisingly, in some cases the competitive ratio is dramatically better than that of the corresponding single server problem.

1 Introduction

Dial-a-ride problems form an important general class of transportation problems. In a dial-a-ride problem, a number of vehicles (servers) move in a metric space in order to process transportation requests; every request for a *ride*, consists of two points of the space, a source and a destination, which must be visited in this order. The problem is to find transportation schedules for the servers that together serve all the given rides, while meeting some optimality criterion.

We study dial-a-ride problems in their natural *online* version, where decisions have to be taken without having any information about future requests. They arrive while processing previous requests. Examples of such problems are taxi-services, elevator systems, drugs courier services, pizza delivery. The online model that we employ is often called the *real time* model, in contrast to the *one-by-one* model, which is the more common model in texts about online optimization [5], but inadequate for dial-a-ride problems. The same real time model is also the natural model and indeed is used for machine scheduling problems [17]. In fact,

* Work of the first author was partly supported by the Dutch Ministry of Education, Culture and Science through a Huygens scholarship.
** Work of the third author was partly supported by MRT Network ADONET of the European Community (MRTN-CT-2003-504438) and the Dutch BSIK/BRICKS project.

many of the algorithms for online dial-a-ride problems are adaptations of online machine scheduling algorithms.

Competitive analysis [5] has become the standard way to study online optimization problems: an online algorithm A is said to be *c-competitive* if, for any instance σ, the cost of A on σ is at most c times the offline optimum on σ. This worst-case measure can be seen as the outcome of a game between the online algorithm and an offline *adversary*, that is trying to build input instances for which the cost ratio is as large as possible. Thus, the emphasis is not, as in offline optimization, on computational complexity, but rather on the amount of inefficiency generated by the absence of complete information.

There is an abundant amount of literature on off-line dial-a-ride problems. A classification effort of the offline complexity of the many variants of dial-a-ride problems has been carried out in [8]. Online single server dial-a-ride problems have a recent but growing literature. The first paper by Ausiello et al. [3] introduced the model for the online traveling salesman problem, which can be seen as a dial-a-ride problem in which the source and destination of the rides coincide. Later works investigated competitiveness of more general dial-a-ride problems [1, 9] and studied different objective functions or different adversarial models [2, 4, 11, 13, 14, 16]. A summary of single server results is contained in the thesis [15].

Prior to this publication, there was essentially no work on online multi-server dial-a-ride problems, except for some isolated algorithms [1, 4]. We give competitive algorithms and negative results for online dial-a-ride problems with any number of servers, with the objective of minimizing either *makespan* or *average completion time*. In the case of makespan we consider two more variants, depending if the servers are required to return in the origin after serving all requests or not. Apart from being the first paper dedicated to multi-server online dial-a-ride problems the results are quite unexpected. The paper gives the first results of online problems for which multiple server versions admit lower competitive ratios than their single server counterparts. In almost all versions of the problem we study here the competitive ratio does not increase. For several versions we see a decrease with respect to the single server version. In some cases competitive ratios even tend to 1 with increasing number of servers.

In scheduling a lot of research has been conducted to online multiple machine problems [17]. In the *one-by-one* model competitive ratios increase with increasing number of machines. In *real time* online scheduling nobody has been able to show smaller competitive ratios for multiple machine problems than for the single machine versions, though here lower bounds do not exclude that such results exist (and indeed people believe they do) [6, 7].

The rest of our paper is structured as follows. In Section 2, we specify and formalize the problems that we study. The competitiveness results are presented in Sections 3, 4, and 5, grouped according to objective function. In Section 6 we give our conclusions.

2 The model

We assume a real time online model, in which requests arrive over time in a metric space M. Every *request* is a triple $(r, x_s, x_d) \in \mathbb{R}_+ \times M \times M$ where r is the *release time* of the request, x_s the location of the source of the request and x_d that of the destination; the pair (x_s, x_d) is called a *ride*. All the information on a request with release time r is revealed only at time r.

An algorithm controls k vehicles or *servers*. Initially, at time 0, all these servers are located in a distinguished point o of the metric space, the origin. The algorithm can then move the servers around the space at speed at most 1. (We do not consider the case in which servers have different maximum speeds; in compliance with machine scheduling vocabulary we could say that the servers are identical and work in parallel.) To process a request, a server has to visit first the source and then the destination of the associated ride. In this paper we only consider problems in which the vehicles have unit capacity, that is, each of them can serve only one ride at a time, and in which rides cannot be preempted, i.e., once started, the destination must be reached before the vehicle can move freely again.

We consider so-called *path metric* spaces, in which the distance d between two points is equal to the length of the shortest path between them. We also require the spaces to be continuous, in the sense that for every $x, y \in M$ and every $a \in [0, 1]$ there is $z \in M$ such that $d(x, z) = ad(x, y)$ and $d(z, y) = (1 - a)d(x, y)$. A discrete space, like a weighted graph, can be extended to a continuous path metric space in the natural way; the continuous space thus obtained is said to be *induced* by the original space. For completeness, we recall that a function $d : M^2 \to \mathbb{R}_+$ is a *metric* if it has the following properties: definiteness ($\forall x, y \in M$, $d(x, y) = 0 \Leftrightarrow x = y$); symmetry ($\forall x, y \in M$, $d(x, y) = d(y, x)$); triangle inequality ($\forall x, y, z \in M$, $d(x, z) + d(z, y) \geq d(x, y)$). When referring to a *general space*, we mean any element of our class of continuous, path-metric spaces. We will also be interested in special cases, namely the real line $\mathbb{R}$ and the real halfline $\mathbb{R}_+$, both with the origin o at 0.

We study as objective functions the *makespan* and the *average completion time*. Defining the *completion time* of a ride as the time at which the ride has been finished at the destination, the makespan is the maximum completion time. Minimizing the makespan is the objective of what we call *nomadic* dial-a-ride problems (N-DARP). The variations in which the goal is minimizing the time at which the last server has returned to the origin after all the requests have been completed are called *homing* dial-a-ride problems (H-DARP)[4]. Also in this case we speak of the objective as the makespan. It will always be clear of the context which of the two notions of makespan we mean. Minimizing average completion time is the objective of the so-called *latency* dial-a-ride problems (L-DARP). When the source and destination of every ride coincide, these problems are called respectively the nomadic traveling salesman problem (N-TSP), the homing traveling salesman problem (H-TSP), and the traveling repairman

[4] Notice that an online algorithm does not know when all requests have been released.

Table 1. Homing TSP competitiveness bounds

	1-server LB	Ref.	1-server UB	Ref.
General spaces	2	[3]	2	[1, 3]
Line	1.64	[3]	1.64	[15]
Halfline	1.5	[4]	1.5	[4]

	k-server LB	Ref.	k-server UB	Ref.
General spaces	2	Thm. 3.1	2	[1]
Line	1.5	↓	1.5	Thm. 3.3
Halfline	1.5	Thm. 3.2	1.5	Thm. 3.2

problem (TRP). In these cases a request is represented by a pair (r, x), with r release time and x location of the point to be visited.

We will use σ to denote a sequence of requests. Given a sequence σ, a feasible *schedule* for σ is a sequence of moves of the servers such that the following conditions are satisfied: (1) the servers start in the origin o, (2) each ride requested in σ is served, (3) the servers do not start to serve any ride before its release time, and (4) in the homing case, the servers end in the origin o. For an online algorithm $\textsc{ol}$, $\textsc{ol}(\sigma)$ will be the cost on σ, and $\textsc{opt}(\sigma)$ the optimal offline cost on σ. $\textsc{ol}$ is said to be c-competitive if $\textsc{ol}(\sigma) \leq c \cdot \textsc{opt}(\sigma)\ \forall \sigma$.

We use $s_1, \ldots, s_k$ to denote the k vehicles, and write $s_j(t)$ for the position of vehicle s_j at time t. For a schedule S of the requests over the vehicles, we denote the total length of S by $|S|$.

3 Homing dial-a-ride problems

We study the Homing k-server TSP and the Homing k-server DARP, shortly H-kTSP and H-kDARP. The results are summarized in Tables 1 and 2, respectively. LB and UB stand for lower bounds and upper bounds on the competitive ratio of the problem, respectively; a lower bound of b means that no algorithm is c-competitive with $c < b$, while an upper bound of c means that there exists a c-competitive algorithm. The tables include references for the results, either to theorems in this paper or to the literature. An arrow in the references points to the more general case from which the corresponding result follows for upper bounds or to a specific case from which it follows for lower bounds.

The lower bound for H-kTSP and H-kDARP on general metric spaces comes from a generalization of the H-kTSP-bound in case $k = 1$ [3, 15].

Theorem 3.1. *Any ρ-competitive algorithm for H-kTSP on general spaces has $\rho \geq 2$.*

Proof. The proof is given in the Appendix. □

A best possible 2-competitive algorithm for the H-kTSP and H-kDARP on general spaces was already known [1].

Table 2. Homing DARP competitiveness bounds

	1-server LB	Ref.	1-server UB	Ref.
General spaces	2	[3]	2	[1]
Line	1.75	[15]	2	↑
Halfline	1.7	[1]	2	↑

	k-server LB	Ref.	k-server UB	Ref.
General spaces	2	Thm. 3.1	2	[1]
Line	1.5	↓	2	↑
Halfline	1.5	Thm. 3.2	2	↑

For the bounds for H-kTSP on the line $\mathbb{R}$ and the halfline $\mathbb{R}_+$ with o in 0, in Table 1, the following property is crucial.

Lemma 3.1. *Let A_k be an algorithm for H-kTSP on $\mathbb{R}_+$. Then there exists an algorithm A_1 for Homing 1-server TSP on $\mathbb{R}_+$ such that $A_1(\sigma) = A_k(\sigma)$ for all σ. Moreover, if A_k is online, A_1 is also online.*

Proof. Move A_1's server s so that $s(t) = \max\{s_1(t), s_2(t), \ldots, s_k(t)\}$ at any time t. All the requests that are served by any s_j will also be served by s, and the value of the makespan is the same for both algorithms. □

This means that for the Homing TSP on the halfline, any algorithm restricted to use only one server does not lose any power. In particular it implies that the 3/2 best possible competitive ratio for the single-server problem is also attainable for H-kTSP by the same algorithm [4], which we give here for sake of completeness.

Algorithm 1 (Move Right If Necessary). We call *right* the direction of increasing (absolute) coordinates. The server moves right at full speed if there are unserved requests to his right. Else, the server moves towards o at full speed.

Theorem 3.2. *Any ρ-competitive algorithm for the H-kTSP on the halfline has $\rho \geq 3/2$. Move Right If Necessary is 3/2-competitive for H-kTSP.* □

The obvious idea for H-kTSP on $\mathbb{R}$, with $k \geq 2$ is to use only 2 servers, one on each of the two halflines $\mathbb{R}_+$ and $\mathbb{R}_-$. That this is true for the optimal offline solution was already noticed by Blom et al. [4].

Lemma 3.2 ([4]). *There is an optimal offline strategy for the Homing k-server TSP on the real line with $k \geq 2$ servers such that no server ever crosses the origin.* □

Algorithm 2 (Split Move Right if Necessary). Assign one server to each halfline. Apply Move Right if Necessary to each halfline.

Theorem 3.3 ([4]). *Split Move Right if Necessary is a best possible 3/2-competitive algorithm for H-kTSP on the real line with $k \geq 2$ servers.* □

Table 3. Nomadic TSP competitiveness bounds

	1-server LB	Ref.	1-server UB	Ref.
General spaces	2.03	↓	2.42	[15]
Line	2.03	[15]	2.06	[15]
Halfline	1.63	[15]	2.06	[15]

	k-server LB	Ref.	k-server UB	Ref.
General spaces	2	Thm. 4.1	2.42	Thm. 4.2
Line	$1+1/k$	Thm. 4.4	$1+O(\log k/k)$	Thm. 4.5
Halfline	$1+1/2k$	Thm. 4.4	$1+O(\log k/k)$	Thm. 4.3

Table 4. Nomadic DARP competitiveness bounds

	1-server LB	Ref.	1-server UB	Ref.
General spaces	2.03	↓	2.62	[15]
Line	2.03	[15]	2.62	↑
Halfline	1.90	[15]	2.62	↑

	k-server LB	Ref.	k-server UB	Ref.
General spaces	2	↓	3.42	[12]
Line	2	Thm. 4.6	3.42	↑
Halfline	1.5	Thm. 4.6	3.42	↑

We notice that the single-server case on the real line, admits a best possible algorithm with competitive ratio $(9+\sqrt{17})/8 \simeq 1.64$ [15].

The lower bounds for H-kDARP in Table 2 come directly from those for H-kTSP, and the upper bounds descend from the algorithm on general spaces [1]. It is worth trying to improve these results on the line and the halfline, similar to the single server case.

4 Nomadic dial-a-ride problems

The competitiveness results for the N-kTSP and the N-kDARP, summarized in Tables 3 and 4, respectively, are more surprising than their counterparts for the Homing versions.

4.1 The nomadic traveling salesman problem

A lower bound of 2 on the competitive ratio of any online algorithm for the N-kTSP on general spaces is easily proved.

Theorem 4.1. *Any ρ-competitive algorithm for the N-kTSP on general spaces has $\rho \geq 2$.*

Proof. Consider the metric space induced by a star graph with $k+1$ rays. At time 1, there must be a ray on which there are no servers. The adversary gives a request on this ray at distance 1 from the origin, serving it immediately. The makespan of the online servers will be at least 2. $\square$

The best algorithm we have found for general spaces is a generalization of a $(1+\sqrt{2})$-competitive algorithm for the single-server case [15].

Algorithm 3 (Return Home(α)). As soon as a new request arrives, let all servers return to the origin at full speed. Once they are all at the origin, compute a set of k paths $\{P_1, \ldots, P_k\}$ serving all requests and having minimum maximum length. For all $j = 1, \ldots, k$ let server s_j start to follow path P_j at the highest possible speed but remaining at a distance at most αt from the origin at any time t.

Theorem 4.2. *Return Home(α) is* $\max\{2+\alpha, \frac{1}{\alpha}\}$*-competitive for N-$k$TSP on general spaces. Hence, Return Home($\sqrt{2}-1$) is* $(1+\sqrt{2})$*-competitive.* $\square$

Having more servers has an enormous impact on the competitiveness of the N-kTSP in case the metric space is the halfline or the line. More precisely, with respect to k, the best competitive ratio we found is $1 + o(1)$ comparing to a 1.63 [15] lower bound for the single server case.

Algorithm 4 (Geometric Progression Speeds). As a preprocessing step, the algorithm delays each request (r, x) for which $x \geq t$ to time x; that is, the release time of each request (r, x) is reset at $\max\{r, x\}$.

Then, let f_k be the unique root greater than 1 of the equation $f_k^k \frac{f_k - 1}{f_k + 1} = 1$, and define $\alpha_j = f_k^{j-1} \frac{f_k - 1}{f_k + 1}$ for $j \in \{1, 2, \ldots, k\}$. For every j, server s_j departs at time 0 from the origin at speed α_j. Servers s_2 to s_k always proceed in the same direction. The first server, instead, turns back at full speed as soon as a new request is given between the origin and its current position, and proceeds until it reaches the unserved request that is nearest to the origin. Then it proceeds away from the origin at full speed until it reaches again the space-time line $s_1(t) = \alpha_1 t$. At that point it follows that space-time line until the next request, turning back as before if necessary.

Theorem 4.3. *Geometric Progression Speeds is f_k-competitive for N-kTSP on the halfline for any $k \geq 1$.*

Proof. First, notice that the modified release times are still lower bounds on the cost of an optimal solution. Thus it is enough to prove that, for every request, the time at which it is served is at most $f_k r$, where r is the modified release time of some (not necessarily the same) request.

For $1 < j < k$, we say that a request (r, x) is in *zone j* if $\alpha_j \leq x/r < \alpha_{j+1}$ (wlog we assume $t > 0$). A request is in zone 1 if $x/r < \alpha_2$; it is in zone k if $x/r \geq \alpha_k$. Thus, every request is in some zone and a request in zone j will be eventually served by server s_j.

For a request (r, x) in a zone j with $1 < j < k$, since the request is served by server s_j at time x/α_j and since $x \leq \alpha_{j+1} r$, the ratio between completion time and release time is at most α_{j+1}/α_j. Similarly, for a request in zone k, since $x \leq r$, the ratio between completion time and release time is at most $1/\alpha_k$.

We still need to give a bound for requests in zone 1. Consider such a request (r, x) which is served at time τ. If $\tau \leq x/\alpha_1$, we know that $\tau \leq (\alpha_2/\alpha_1)r$. If $\tau > x/\alpha_1$, it means that the first server, at time x/α_1, was either going to serve some other request between its current position and the origin, or was returning after having served such a request. Either way, if the release time of that request is $r' \geq r$, it has to be at least $r' \geq \tau/(1 + \alpha_1 + \alpha_2)$, since otherwise at time at most $r' + \alpha_1 r' + \alpha_2 r' < \tau$ the server would have returned and served request (r, x). Similarly, if $r' < r$ we have that $\tau \leq r' + \alpha_1 r' + \alpha_2 r < (1 + \alpha_1 + \alpha_2)r$.

Thus, the competitive ratio is bounded by

$$\begin{aligned}
&\max\{1 + \alpha_1 + \alpha_2,\ \alpha_2/\alpha_1,\ \alpha_3/\alpha_2,\ \ldots,\ \alpha_k/\alpha_{k-1},\ 1/\alpha_k\} \\
&= \max\{1 + \frac{f_k - 1}{f_k + 1} + f_k \frac{f_k - 1}{f_k + 1},\quad f_k,\quad \frac{f_k + 1}{f_k^{k-1}(f_k - 1)}\} \\
&= f_k.
\end{aligned}$$

□

The following Lemma gives insight into the quality of the bound f_k.

Lemma 4.1. *For every $k \geq 1$, $f_k \leq 1 + \frac{2\log k + 2}{k}$.*

Proof. The proof is given in the Appendix. □

The above upper bound compares with a lower bound of $1 + \frac{1}{2k}$ for the problem on the halfline.

Theorem 4.4. *Any ρ-competitive algorithm for N-kTSP on the halfline has $\rho \geq 1 + \frac{1}{2k}$. Any ρ-competitive algorithm for N-kTSP on the line has $\rho \geq 1 + \frac{1}{k}$.*

Proof. The proof is given in the Appendix. □

By contrast, the best known algorithm for a single server on the halfline has a competitive ratio of 2.06 [15].

For the line, a simple idea is to split the servers as evenly as possible on the two halflines and then use Geometric Progression Speeds.

Algorithm 5 (Split Geometric Progression Speeds). Assign $\lceil k/2 \rceil$ servers to $\mathbb{R}_+$ and $\lfloor k/2 \rfloor$ servers to $\mathbb{R}_-$, and apply Algorithm 4 to each of them.

Theorem 4.5. *Split Geometric Progression Speeds is $f_{\lfloor k/2 \rfloor}$-competitive for N-kTSP on the line for any $k \geq 2$.*

Proof. Notice that the only lower bounds on the offline cost that we used in the proof of Theorem 4.3 were simply the distance of every request from the origin and the release time of every request, which are valid independently of the number of vehicles of the offline algorithm. In particular, they hold even if the number of offline vehicles is twice the number of online vehicles. Thus, we can analyze the competitiveness of the online servers on each of the two halflines separately and take the worst of the two competitive ratios. □

4.2 The nomadic dial-a-ride problem

For the single server Nomadic Dial-A-Ride problem, the best known algorithm has a competitive ratio of $\frac{3+\sqrt{5}}{2} \simeq 2.618$ [15]. For the multiple server case, we were not able to reach this competitive ratio. However, we can prove that an adaption of the 2-competitive algorithm for H-kDARP [1] is $(2+\sqrt{2})$-competitive in a way similar to [12] where this was proved to hold for the single server problem.

In contrast with N-kTSP, here even on the line and the halfline the competitive ratio cannot approach 1 as the number of vehicles grows.

Theorem 4.6. *Any ρ-competitive algorithm for N-kDARP on the line has $\rho \geq 2$. Any ρ-competitive algorithm for N-kDARP on the halfline has $\rho \geq 3/2$.*

Proof. The proof is given in the Appendix. □

Thus, the lower bounds for multiple servers are not very far from those for the single server cases, which are 2.03 for the line and 1.9 for the halfline [15]. We have not tried particularly hard to design algorithms for N-kDARP on these two metric spaces.

5 Latency dial-a-ride problems

The results on the k-server TRP and the Latency k-server DARP, shortly kTRP and L-kDARP, are summarized in Tables 5 and 6, respectively.

First we study kTRP. A $(1+\sqrt{2})^2$-competitive algorithm for kTRP follows from a more general result on L-kDARP in Theorem 5.4.

All the lower bounds given for the N-kTSP also apply to the kTRP because the adversarial sequences consisted of a single request.

Theorem 5.1. *Any ρ-competitive algorithm for kTRP has $\rho \geq 2$ on general spaces, $\rho \geq 1 + 1/k$ on the line and $\rho \geq 1 + 1/2k$ on the halfline.* □

Interestingly, for kTRP on the halfline an approach similar to that for N-kTSP can be used to get a competitive ratio of $1 + o(1)$.

Algorithm 6 (Geometric Progression Speeds with Sweeps). As a preprocessing step, the algorithm delays every request (r, x) for which $x \geq r$ to time x; that is, the release time of each request (r, x) is reset at $r' := \max\{r, x\}$.

Then, let g_k be the unique root greater than 1 of the equation $g_k^k = \frac{3g_k-1}{g_k-1}$, and define $\alpha_j = g_k^{j-k-1}$ for $j \in \{2, 3, \ldots, k\}$. For every $j > 1$, server s_j departs at time 0 from the origin at speed α_j and never turns back. The first server s_1 waits in the origin until the first request (r_0, x_0) is released with $x_0 < s_2(r_0)$. For $i \geq 0$, define $t_i = g_k^i r_0$. For $i \geq 1$ during the interval $[t_{i-1}, t_i]$, s_1 moves first from o to $\frac{g_k-1}{2}t_{i-1}$ and back to o at full speed.

Theorem 5.2. *Algorithm 6 is g_k-competitive for kTRP on the halfline.*

Table 5. TRP competitiveness bounds

	1-server LB	Ref.	1-server UB	Ref.
General spaces	2.41	↓	5.83	[13]
Line	2.41	[9]	5.83	↑
Halfline	2	[15]	3.5	[15]

	k-server LB	Ref.	k-server UB	Ref.
General spaces	2	Thm. 5.1	5.83	Thm. 5.4
Line	$1+1/k$	Thm. 5.1	$1+O(\log k/k)$	Thm. 5.3
Halfline	$1+1/2k$	Thm. 5.1	$1+O(\log k/k)$	Thm. 5.2

Proof. As in the proof of Theorem 4.3 we define zones $\{(r', x) \mid \alpha_j \leq x/r' < \alpha_{j+1}\}$ for $0 < j < k$ (using $\alpha_{k+1} = \infty$). The proof for requests in zones 2 to k is the same as in the proof of Theorem 4.3.

Take any request in zone 1, i.e., a request (r', x) such that $x < \alpha_2 r'$ and suppose it served during at time $\tau \in [t_{i-1}, t_i]$ for some i. If $r' \geq t_{i-1}$, then since $\tau \leq t_i$ the ratio between τ and r' is at most g_k by definition of t_i, $i \geq 0$.

If $r' < t_{i-1}$, then since $\tau > t_{i-1}$ only two possible cases remain. First, the situation that $x > \frac{g_k - 1}{2} t_{i-2}$. Since $\tau = t_{i-1} + x$ and $r' \geq x/\alpha_2$, we have

$$\frac{\tau}{r'} \leq \frac{x + t_{i-1}}{x/\alpha_2} \leq \alpha_2 \left(1 + \frac{2g_k t_{i-2}}{(g_k - 1)t_{i-2}}\right) = \alpha_2 \frac{3g_k - 1}{g_k - 1} \leq \alpha_2 g_k \leq g_k.$$

In the second situation, $x \leq \frac{g_k - 1}{2} t_{i-2}$. r' must be such that s_1 was already on his way back to 0 during $[t_{i-2}, t_{i_1}]$, in particular $r' \geq g_k t_{i-2} - x$. Thus,

$$\tau/r' \leq \frac{g_k t_{i-2} + x}{g_k t_{i-2} - x} \leq \frac{3g_k - 1}{g_k + 1} \leq g_k. \qquad \square$$

Lemma 5.1. *For every $k \geq 1$, $g_k \leq 1 + \frac{2\log k + 3}{k}$.*

Proof. The proof is along the same lines as the proof of Lemma 4.1. $\square$

For kTRP on the line, as for N-kTSP on the line, we can split the k servers evenly between the two halflines.

Algorithm 7 (Split Geometric Progression Speeds with Sweeps). Assign $\lceil k/2 \rceil$ servers to $\mathbb{R}_+$ and $\lfloor k/2 \rfloor$ servers to $\mathbb{R}_-$, and apply Algorithm 6 to each of them.

Theorem 5.3. *Algorithm 7 is $g_{\lfloor k/2 \rfloor}$-competitive for kTRP on the line.*

Proof. As in the proof of Theorem 4.5, the lower bounds on the optimal completion times are valid independently of the number of offline vehicles. $\square$

Table 6. Latency DARP competitiveness bounds

	1-server LB	Ref.	1-server UB	Ref.
General spaces	3	↓	5.83	[13]
Line	3	[9]	5.83	↑
Halfline	2.41	[15]	5.83	↑

	k-server LB	Ref.	k-server UB	Ref.
General spaces	2	Thm. 5.1	5.83	Thm. 5.4
Line	1.5	Thm. 5.5	5.83	↑
Halfline	1.33	Thm. 5.5	5.83	↑

We turn to L-kDARP now. The best known algorithm for the single server latency dial-a-ride problem, called Interval [13], is an adaptation to vehicle routing of an algorithm for online job scheduling [10]. A rather straightforward generalization of Interval has the same competitive ratio as the one for the single server [13].

Theorem 5.4. *A multiple server adaption of Interval is* $(1+\sqrt{2})^2$*-competitive for L-kDARP.* □

As in the case of N-kDARP, the competitive ratio cannot approach 1 when k grows. The lower bounds that we are able to prove are weaker though.

Theorem 5.5. *Any* ρ*-competitive algorithm for L-kDARP on the line has* $\rho \geq 3/2$*. Any* ρ*-competitive algorithm for L-kDARP on the halfline has* $\rho \geq 4/3$*.*

Proof. At time 1, consider the barycentrum $\overline{s} = \frac{1}{k}\sum_i s_i$ of the positions of the servers; assume wlog $\overline{s} \leq 0$. An adversary gives k requests $(1,1,0)$. Since the time needed to serve a ride and return to point 1 is 2, we may assume that each of these rides will be handled by a different server. Thus, the online cost is at least $\sum_i(1+1-s_i+1) = 3k - k\overline{s} \geq 3k$, compared to an optimal offline cost of $2k$. The proof for the halfline is similar. □

6 Conclusions

After analyzing the differences between multiple and single vehicle variants we can conclude that, generally speaking, having multiple vehicles is more beneficial to the online algorithm than to the offline adversary. With one exception (the nomadic dial-a-ride problem), the algorithms for the multiple server case have a competitive ratio that is at least as good as for a single server. In some cases, including the traveling repairman problem on the line, it is even possible to do better and approach the offline cost when there are enough vehicles. In the presence of proper rides, these extremely favorable situation cannot occur. Still in N-kDARP and L-kDARP it is conceivable that the competitive ratios become lower than those of the corresponding single vehicle problems.

References

[1] N. Ascheuer, S. O. Krumke, and J. Rambau. Online dial-a-ride problems: Minimizing the completion time. In H. Reichel and S. Tison, editors, *Proc. 17th Symp. on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 639–650. Springer-Verlag, 2000.

[2] G. Ausiello, V. Bonifaci, and L. Laura. The on-line asymmetric traveling salesman problem. In F. Dehne, A. López-Ortiz, and J. Sack, editors, *Proc. 9th Workshop on Algorithms and Data Structures*, volume 3608 of *Lecture Notes in Computer Science*, pages 306–317. Springer-Verlag, 2005.

[3] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.

[4] M. Blom, S. O. Krumke, W. E. de Paepe, and L. Stougie. The online TSP against fair adversaries. *INFORMS Journal on Computing*, 13(2):138–148, 2001.

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

[6] B. Chen and A. P. A. Vestjens. Scheduling on identical machines: How good is LPT in an on-line setting? *Operations Research Letters*, 21:165–169, 1998.

[7] J. R. Correa and M. R. Wagner. LP-based online scheduling: From single to parallel machines. In *Integer programming and combinatorial optimization*, volume 3509 of *Lecture Notes in Computer Science*, pages 196–209. Springer-Verlag, 2005.

[8] W. E. de Paepe, J. K. Lenstra, J. Sgall, R. A. Sitters, and L. Stougie. Computer-aided complexity classification of dial-a-ride problems. *INFORMS Journal on Computing*, 16(2):120–132, 2004.

[9] E. Feuerstein and L. Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268(1):91–105, 2001.

[10] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3):513–544, 1997.

[11] D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. In G. Bongiovanni, G. Gambosi, and R. Petreschi, editors, *Proc. 4th Italian Conference on Algorithms and Complexity*, volume 1767 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag, 2000.

[12] S. O. Krumke. Online optimization: Competitive analysis and beyond. Habilitation Thesis, Technical University of Berlin, 2001.

[13] S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. *Theoretical Computer Science*, 295(1-3):279–294, 2003.

[14] S. O. Krumke, L. Laura, M. Lipmann, A. Marchetti-Spaccamela, W. E. de Paepe, D. Poensgen, and L. Stougie. Non-abusiveness helps: an $O(1)$-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem. In K. Jansen, S. Leonardi, and V. V. Vazirani, editors, *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 2462 of *Lecture Notes in Computer Science*, pages 200–214. Springer-Verlag, 2002.

[15] M. Lipmann. *On-Line Routing.* PhD thesis, Eindhoven University of Technology, 2003.

[16] M. Lipmann, X. Lu, W. E. de Paepe, R. A. Sitters, and L. Stougie. On-line dial-a-ride problems under a restricted information model. *Algorithmica*, 40(4):319–329, 2004.

[17] J. Sgall. On-line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 196–231. Springer, 1998.

Appendix

Proof (of Theorem 3.1). Consider the metric space induced by a star graph with kn rays (for some fixed n); i.e., the space induced by $M = (X, d)$ where $X = \{o, 1, 2, \ldots, kn\}$, $d(o, x) = 1$ for all $x \in X \setminus \{o\}$ and $d(x, y) = 2$ for all $x, y \in X \setminus \{o\}$ such that $x \neq y$.

At time 0, an adversary gives a request in each of the kn points of $X \setminus \{o\}$. Moreover, if one of the servers of an online algorithm visits a point $i \in X \setminus \{o\}$ at time t when $t \leq 2n - 2$, then at time $t + 1$ a new request in point i is released. In this way, until time $2n - 2$ the online algorithm keeps facing requests in all kn locations. The makespan will be at least $2n - 2 + 2n - 1 = 4n - 3$.

Instead, the adversary can serve the requests in first-in first-out order, ignoring old requests at repeatedly requested locations. Hence, at most $k(n - \lfloor t/2 \rfloor)$ requests are yet to be served at any time $t \leq 2n - 2$, and the servers can finish by time $2n$. Thus, the competitive ratio is at least $(4n - 3)/2n$, which can be made arbitrarily close to 2. □

Proof (of Lemma 4.1). We consider the unique root greater than 1 of the equation $z^k = 1 + \frac{2}{z-1}$. Since as z tends to infinity the left hand side is greater than the right hand side, and the root is unique, it suffices to prove that $Z = 1 + \frac{2 \log k + 2}{k}$ satisfies $Z^k \geq 1 + \frac{2}{Z-1}$. By the binomial theorem,

$$Z^k - 1 = \sum_{j=1}^{k} \binom{k}{j} \frac{(2 \log k + 2)^j}{k^j}$$

and this, using the standard fact that $\binom{k}{j} \geq \frac{k^j}{j^j}$, is at least

$$\sum_{j=1}^{k} \frac{(2 \log k + 2)^j}{j^j} \geq \sum_{j=1}^{\lfloor \log k \rfloor + 1} \left(\frac{2 \log k + 2}{j} \right)^j \geq \sum_{j=1}^{\lfloor \log k \rfloor + 1} 2^j$$
$$\geq 2^{\log k + 1} - 2 = 2k - 2.$$

Now it can be verified that for all $k > 1$, $2k - 2 > \frac{2k}{k + 2 \log k + 2} = 1 + \frac{2}{Z-1}$. Finally, the bound also holds for $k = 1$ since $f_1 = 1 + \sqrt{2} \leq 3$. □

Proof (of Theorem 4.4). At time 1, consider the positions of the servers and the points 0 and 1. These $k + 2$ points induce a partitioning of $[0, 1]$ into (at most) $k + 1$ subintervals. Call an interval *external* if it contains 0 or 1. If there is an external interval of length at least $1/2k$, the adversary gives a request in the extreme of the interval so that the online algorithm cannot serve the request before time $1 + 1/2k$. Otherwise, there must be an internal interval of length at least $\frac{1-1/k}{k-1} = 1/k$. The adversary then gives a request in the middle of this interval, and the online algorithm cannot serve it before time $1 + 1/2k$. In both cases the adversary will serve the request immediately at time 1.

The proof for the case of the line is similar. □

Proof (of Theorem 4.6). At time 1, consider the online server nearest to an extreme of $[-1, 1]$. Suppose that this server is located at x and that wlog $x \geq 0$. The adversary then requests k rides from -1 to $x - 1$. Since there are no servers in the interval $[-1, -x)$, the latest completion time of the online algorithm will be at least $1 + x + (1 + x)$ while the adversary can pay only $1 + x$.

The proof for the case of the halfline is similar. □