ELSEVIER Theoretical Computer Science 268 (2001) 91–105

Theoretical Computer Science

www.elsevier.com/locate/tcs

On-line single-server dial-a-ride problems

Esteban Feuerstein[a,1], Leen Stougie[b,c,*,2]

[a]Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, and Instituto de Ciencias, Universidad de General Sarmiento, Argentina

[b]Combinatorial Optimization Group, Faculty of Mathematics, Eindhoven University of Technology, P.O. Box 513, 5600MB Eindhoven, The Netherlands

[c]Centrum voor Wiskunde en Informatica (CWI), P.O. Box 94079, 1090GB Amsterdam, The Netherlands

Accepted May 2000

Abstract

In this paper results on the dial-a-ride problem with a single server are presented. Requests for rides consist of two points in a metric space, a source and a destination. A ride has to be made by the server from the source to the destination. The server travels at unit speed in the metric space and the objective is to minimize some function of the delivery times at the destinations. We study this problem in the natural on-line setting. Calls for rides come in while the server is traveling. This models, e.g. the taxi problem, or, if the server has capacity more than 1 a minibus or courier service problem. For the version of this problem in which the server has infinite capacity having as objective minimization of the time the last destination is served, we design an algorithm that has competitive ratio 2. We also show that this is best possible, since no algorithm can have competitive ratio better than 2 independent of the capacity of the server. Besides, we give a simple 2.5-competitive algorithm for the case with finite capacity. Then we study the on-line problem with objective minimization of the sum of completion times of the rides. We prove a lower bound on the competitive ratio of any algorithm of $1 + \sqrt{2}$ for a server with any capacity and of 3 for a server with capacity 1. Finally, we present the first competitive algorithm for the case the server has infinite capacity and the metric space is the real line. The algorithm has competitive ratio 15. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Dial-a-ride; On-line optimization; Competitive analysis

* Corresponding author.

E-mail addresses: efeuerst@dc.uba.ar (E. Feuerstein), leen@win.tue.nl (L. Stougie).

[1] Partly supported by the KIT program of the European Community under contract n. 131 (DYNDATA), by UBA project EX070/J "Algoritmos eficientes para problemas on-line con aplicaciones" and UBA project EX209 "Modelos y técnicas de optimización combinatoria".

[2] Supported by the TMR Network DONET of the European Community ERB TMRX-CT98-0202 and by the KIT program of the European Community under contract n. 131 (DYNDATA).

1. Prologue

Dial-a-ride is a name that covers a rich variety of problems. The common characteristics of dial-a-ride problems is that there are servers that travel in some metric space to serve requests for rides. Each ride is given by two points in the metric space, a *source*, which is the starting point of the ride, and a *destination*, which is the end point of the ride. The problem is to assign rides to servers and to route the servers through the metric space such as to meet some optimality criterion.

The variety in dial-a-ride problems comes from characteristics like the number and the capacity of servers, existence or not of time-windows on the requests, the type of metric space and the particular objective function. In [17] a classification for dial-a-ride problems has been proposed similar to that developed for scheduling problems in [15].

In this paper we study *on-line* dial-a-ride problems. We present first results in this field of very natural on-line optimization problems that has so far remained virtually unexplored.

Requests for rides come in over time, while the server(s) are enroute serving other rides. This model accommodates practical situations that occur for taxi services in which the capacity of each server is 1, but also for courier services in which the capacity of each server may be regarded as infinite, or in between the situation of minibus services in which the servers have some finite capacity.

We emphasize here the particular character of the on-line setting in which the time flows while decisions are made and executed. Thus, there is a history, a present, and a future. The model is a very natural one in on-line routing and scheduling but has received relatively little research attention in the field of on-line optimization. We could call the model the *real-time* model in contrast to what we could call the *batch* model. In the latter model, tasks arrive one by one and decisions about them have to be made immediately and irrevocably. The execution of the resulting planning is done after the whole planning is made.

The essential feature of the real-time model, which is missing in the batch model, is that waiting, or postponing decisions, is allowed, at a cost that depends on the time that elapses, and can actually be beneficial. In the batch model this does not make any sense. In the batch model decisions are irrevocable. In the real-time model only the history is irrevocable.

In the literature some studies on on-line execution problems exist, so far in the context of scheduling [18, 21]. As far as we know, the only paper on on-line routing within this model is [4], studying the on-line traveling salesman problem.

We will design *deterministic* algorithms for a family of dial-a-ride problems with a single server on a metric space that satisfies some general conditions exposed at the end of this section. The algorithms will be analysed on their *competitive ratio*, i.e., the worst-case ratio between the objective value produced by the algorithm and the optimal off-line value.

We first consider problems in which the objective is to minimize the time it takes to serve all the rides and return to the origin. We call this the *makespan* of the service, in analogy to the makespan in scheduling problems. We consider two different cases, in which the server has infinite capacity and in which the server has finite capacity, which includes the special case of unit capacity. A precise description can be found in Section 2.

We show in that section that our algorithms for the cases of finite capacity $k \geqslant 1$ and infinite capacity have competitive ratio 2.5, and 2, respectively. We also provide a lower bound of 2 on the competitive ratio of any deterministic algorithm that holds for both versions of the problem. This implies that the algorithm for the problem with capacity ∞ is *best possible*.

After that in Section 3 we study the dial-a-ride problem with the objective of minimizing the *average completion time* of the requests, also called the *latency*. The completion time of a request is the time that a request is served, i.e., the corresponding ride from source to destination has been completed. The general problem on any metric space is NP-hard (see [6, 17]). The complexity of the problem on the line as metric space is also NP-hard in case of finite capacity [16] and is unknown if the server has infinite capacity and we conjecture that this is NP-hard as well. The only positive result here is on a highly restricted version with one server having capacity 1, moving on the line and there exists a point such that all sources are lying left of this point and all destinations right of it. This version is polynomially solvable [17].

We will prove a lower bound of $1+\sqrt{2}$ on the competitive ratio of any deterministic algorithm for the on-line problem with a single server having any capacity and a lower bound of 3 for the same problem with a server having capacity 1. For the problem on the real line in which the server has infinite capacity we present a first competitive algorithm having competitive ratio 15, leaving a considerable gap with the lower bound. We also prove that the same algorithm is 9-competitive for the particular case in which sources and destinations coincide, which has been referred to in the literature as the deliveryman problem or the Traveling Repairman Problem (TRP).

In the epilogue, we give some conclusions and proposals for a host of interesting problems for future research in the field of dial-a-ride.

In the literature a number of publications have appeared on off-line dial-a-ride problems, and if we include routing and scheduling problems as special cases the literature is abundant. Restricting to “true” dial-a-ride problems, in which each ride is defined by a source and a destination that do not coincide a number of algorithms and heuristics have been presented from an empirical point of view. For an overview of such methods and comparative studies on their empirical performance we refer to [7, 19].

From a theoretical point of view we only know of work concerning the objective of minimizing the time required to complete all rides. Frederickson et al. [12] showed that the problem with a single server having unit capacity on a general metric space is NP-hard. Atallah and Kosaraju [3] gave the first polynomial time algorithm for this problem on the line. Frederickson and Guan showed that the problem is already NP-hard on a tree [11] unless preemption of the rides is allowed in which case it is polynomially solvable [10]. Approximation algorithms for the NP-hard versions of this problem are given in [12, 11].

For a single server with finite capacity greater than one Guan [13] proved that the problem is NP-hard even on the line, unless preemption of the rides is allowed, in which case he presented a polynomial time algorithm improving an old result of Karp [14]. On a tree this problem is NP-hard even if preemption is allowed [13]. Worst-case ratios for approximation algorithms for this problem on the line, on a tree, and on general n-point metric spaces, with and without allowing preemption have been presented recently by Charikar and Raghavachari [8].

As far as we know virtually nothing appeared so far on the on-line dial-a-ride problem. A recent technical report by Ascheuer et al. [1] emphasizes the need from practice for studying on-line combinatorial optimization problems in general and some dial-ride problems in particular. Inspired by such a practical problem Ascheuer et al. [2] are studying dial-a-ride problems, and have independently of us obtained several 2.5-competitive algorithms for the single-server problem with unit capacity minimizing makespan. They differ from ours, although some only slightly.

As a consequence of the work presented in this paper, Feuerstein et al. studied on-line dial-a-ride problems in a multi-threaded environment in [9].

Closely related to the problems considered here is the before-mentioned work on the on-line Traveling Salesman Problem (TSP) [4]. The TSP can be seen as a special version of the dial-a-ride problem in which for each request source and destination coincide. In fact, this relation is exploited in some of the results we derive in this paper.

We finish this section by defining a class $\mathscr{M}$ of metric spaces that will be considered in the rest of the paper. Every metric space in $\mathscr{M}$ must be symmetric, which is usually part of the definition of metric space, i.e., for every pair of points x, y in M, $d(x,y) = d(y,x)$, where $d(x,y)$ denotes the distance from x to y. $\mathscr{M}$ contains all continuous metric spaces, i.e., every metric space M having the property that the shortest path from $x \in M$ to $y \in M$ is continuous, formed by points in M, and has length $d(x,y)$. For continuous metric spaces the times at which a request can be made can be any non-negative real number.

Next, $\mathscr{M}$ contains discrete metric spaces representable by an underlying graph with all edges having unit length. The vertices are the points of the metric space. Working on such spaces time needs to be discretized, i.e., the times t_i at which requests are made are non-negative integers, and the server determines its strategy at integer points in time being at a point in the metric space (vertex of the graph) and either remains there or movev in one time step to a neighbouring point in the metric space.

Thus, an example of a model that we do not consider here is one in which the server moves on a road network of freeways and a request can arrive while he is moving between two exits and he has to proceed to the next exit before being able to change his strategy.

2. Competitiveness of the makespan problem

We will first study the problem in which there is a single server starting at the origin at time 0 that is to serve requests for rides. Each request j is characterized by a pair of points $\langle s_j, d_j \rangle$, with s_j the source and d_j the destination being points in some underlying metric space belonging to the class $\mathscr{M}$ defined at the end of the previous section. Moreover, a ride j has a *release date* r_j. A ride is then to be made from s_j to d_j, in that order, and the ride cannot be started in s_j before time r_j. We will consider the three variations induced by the various capacity restrictions of the server. In the first one, the server has capacity 1, which means that as soon as he has started a ride he must finish it before being able to start any other ride (we do not allow preemption of rides). In the second, he has capacity $k > 1$ implying that he can collect up to k sources before going to any of their destinations. In the last, the capacity of the server is infinite, which means that he can collect any number of sources before getting to their destinations. In all cases the server travels at a speed of at most 1 per time unit. He may also wait. The objective is to find a route in which all requested rides are served and that ends in the origin such that a minimum amount of time is required. We call the time at which the server finishes its route the makespan.

The requests for rides are communicated to the server over time while he may already be serving rides that have been communicated to him before. At the start of the problem nothing about the rides is known, not even their number. The release date of a ride coincides with the time the request for that ride is presented. At that time the source and the destination of that ride become known.

Before presenting algorithms we first state a lower bound on the competitive ratio of any deterministic algorithm for the single-server problem independent of his capacity.

Theorem 2.1. *For the single-server on-line dial-a-ride problem on any metric space belonging to $\mathscr{M}$, minimizing makespan, any deterministic algorithm must have a competitive ratio of at least* 2, *independently of the capacity of the server.*

Proof. The proof is direct from the lower bound of 2 on the competitive ratio of any algorithm for the on-line (TSP) [4, Theorem 3.2]. The on-line TSP is a specific dial-a-ride problem with the property that source and destination of each ride coincide. Therefore, every ride is completed at the moment it is started and the capacity of the server becomes irrelevant. □

Notice that the lower bound holds for any metric space in $\mathscr{M}$. For specific metric spaces, like the real line of the Euclidean plane the lower bound might be smaller.

We first present a straightforward algorithm that works for any capacity. It has competitive ratio 2.5 and is the best known so far for the situation of finite capacity.

The algorithm consists of repeatedly performing the following strategy:

1. Whenever the server is at the origin, it starts to follow an optimal route that serves all the requests for rides yet to be served and goes back to the origin (taking into account the capacity restriction).

We notice that the algorithm ignores all requests that are presented while the server is on the route that he computed last. That is why we call the algorithm DLT (for Don't Listen while Traveling).

This algorithm fits in spirit in the algorithmic framework presented by Shmoys et al. [20] for on-line scheduling minimizing makespan within a real-time model. They propose algorithms that start executing a yet to be scheduled set of jobs at the moment that all machines are available. In this way, the processing of the jobs is done in batches of subsequences of jobs. Each next batch consists of the jobs being presented during the complete execution of the previous batch. The batches are then scheduled off-line. Shmoys et al. [20, Theorem 2.1] says that if for the off-line scheduling of the batches a ρ-approximation algorithm is used then the resulting algorithm for the on-line problem is 2ρ-competitive.

We will prove a similar result here for DLT. Specifically, we will show that DLTρ, defined as DLT but working with ρ-approximate instead of optimal routes, is 2.5 ρ-competitive. That we do not achieve 2ρ-competitiveness here comes from the fact that a ride may be regarded as a job with a set-up time equal to the time it takes to reach the source of the ride. The adversary model we employ allows the adversary to reach the source of a ride before the ride is communicated to the on-line server. In the proof of the following theorem, we will see how this phenomenon causes the extra $\frac{1}{2}\rho$.

The result holds for any metric space. In the proofs of the theorems in this section we use for any route or path T the notation $|T|$ to denote its length. We notice, that given the assumption that the server does not travel faster than unit speed the length of a tour is a lower bound on the time required to traverse it.

Theorem 2.2. *DLTρ has competitive ratio 2.5ρ for the on-line dial-a-ride problem on any metric space with a single server having any capacity and minimizing the makespan.*

Proof. Let t be the time the last request is presented. Obviously, the optimal off-line solution value Z^{OPT} cannot be less than t. If the server is at the origin o at time t, then obviously the length of the tour T_R still to be followed for serving the yet unserved rides is at most ρ times that of the optimal *off-line tour T* serving all the rides: $|T_R| \leqslant \rho |T|$. Moreover, $|T| \leqslant Z^{\mathrm{OPT}}$. Thus, the solution value $Z^{\mathrm{DLT}\rho}$ obtained by DLTρ satisfies $Z^{\mathrm{DLT}\rho} = t + |T_R| \leqslant (1+\rho) Z^{\mathrm{OPT}} \leqslant 2\rho Z^{\mathrm{OPT}}$.

Now, suppose that at time t, DLTρ is on a tour T_C, serving rides in the set C, which is of length at most ρ times the length of the optimal tour T_C^* for serving the rides in C. Let R be the set of requests that have been presented after DLTρ started from this tour at o. Let t' be the time the first ride in R was presented, and let $p^{\mathrm{OPT}}(t') = p^*$ be the position of the optimal off-line server at time t'. Obviously, the optimal off-line server must travel $|T_R^*|$, the length of the optimal tour serving the rides in R, but he might benefit from the fact that at time t' he does not have to go any more from o to p^*. Thus, $Z^{\mathrm{OPT}} \geqslant t' + |T_R^*| - d(o, p^*)$. Moreover, obviously, $Z^{\mathrm{OPT}} \geqslant |T_C^*|$ and $Z^{\mathrm{OPT}} \geqslant 2d(o, p^*)$.

At time t', DLTρ first has to finish the tour T_C, and afterwards a tour T_R, serving all rides in R, having length $|T_R| \leqslant \rho |T_R^*|$. Thus,

$$Z^{\mathrm{DLT}\rho} \leqslant t' + |T_C| + |T_R| \leqslant t' + \rho|T_R^*| + \rho|T_C^*| + d(o, p^*) - d(o, p^*).$$

Since at t' the optimal off-line server cannot be further away from o than t' we have $t' - d(o, p^*) \geqslant 0$. Therefore,

$$Z^{\mathrm{DLT}\rho} \leqslant \rho(t' + |T_R^*| - d(o, p^*)) + \rho|T_C^*| + d(o, p^*). \tag{1}$$

This together with the above lower bounds on Z^{OPT} establishes the competitive ratio of 2.5ρ. □

As a corollary we obtain the competitive ratio of DLT, which works with optimal tours ($\rho = 1$).

Corollary 2.1. *DLT has competitive ratio* 2.5 *for the on-line dial-a-ride problem on any metric space with a single server having any capacity and minimizing the makespan.*

Proof. The proof follows directly from the previous theorem by setting $\rho = 1$. That the ratio is tight for DLT can be seen from the following input sequence on the line as metric space. At time 0, there is a request $\langle \frac{1}{2}, 0 \rangle$, that DLT starts serving immediately. Just before he is back at 0, at time $1 - \varepsilon$, a request $\langle 1 - \varepsilon, 1 \rangle$ is presented, and finally, at time $1 + \varepsilon$ comes a request $\langle 1 + \varepsilon, \frac{1}{2} \rangle$. Note that DLT will do a separate tour for each of the requests, with a total completion time of $1 + 2 + 2 + 2\varepsilon = 5 + 2\varepsilon$. On the other hand, the adversary may go at the beginning towards $1 - \varepsilon$, where he gets just at the time the second request is presented. He serves it and goes to $1 + \varepsilon$, where he serves the third request, arriving at point $\frac{1}{2}$ at time $\frac{3}{2} + 2\varepsilon$. He can then serve the first request, finishing the whole sequence at time $2 + 2\varepsilon$. □

We emphasize that the competitive ratios exhibited in the previous theorems hold for any metric space and not only for spaces of the class $\mathscr{M}$.

For the on-line problem with capacity ∞, we propose a more complicated algorithm, which is 2-competitive and hence best possible. The algorithm determines at each time t the behaviour of the server. It is based on the algorithm that was proposed for the on-line TSP in [4], which has been called PAH (for Plan At Home) there, and in fact has the same competitive ratio. We call the algorithm TIR (for Temporarily Ignore Requests).

We denote the position of the TIR algorithm at time t by p.

1. Whenever the server is at the origin o, it starts to follow an optimal route that serves all the requests for rides yet to be served and goes back to the origin.
2. If at time t a new request is presented with source s and destination d, then it takes one of two actions depending on its current position p, s and d:

2(a) If $d(s,o) > d(p,o)$ or $d(d,o) > d(p,o)$, then the server goes back to the origin (following the shortest path from p) where it appears in a Case 1 situation.

2(b) If both $d(s,o) \leqslant d(p,o)$ and $d(d,o) \leqslant d(p,o)$ then the server ignores the request until it arrives at the origin, where again it reenters Case 1.

Theorem 2.3. *TIR is has competitive ratio* 2 *for the on-line dial-a-ride problem on any metric space in the class* $\mathscr{M}$ *with a single server having infinite capacity and minimizing the makespan. Therefore, this algorithm is best possible.*

Proof. Let t be the time the last request is presented and let that request be the ride $\langle s,d\rangle$. Let p be the position of TIR at time t. Obviously, $Z^{\mathrm{OPT}} \geqslant t$. As in the proof of the previous theorem, let T be an optimal off-line route that makes all rides requested. Obviously, $Z^{\mathrm{OPT}} \geqslant |T|$. We consider each of the three cases described.

Case 1: The server will follow from time t the optimal route making all rides not yet completed. This route is certainly not longer than $|T|$, and therefore we have for TIR's solution value $Z^{\mathrm{TIR}} \leqslant t + |T| \leqslant 2Z^{\mathrm{OPT}}$.

Case 2a: The server will return to the origin and start from there an optimal route making all yet uncompleted rides. Thus, $Z^{\mathrm{TIR}} \leqslant t + d(p,o) + |T|$. In this case, we have for the optimal off-line route a new lower bound, since the ride $\langle s,d\rangle$ cannot be served before time t. Therefore, $Z^{\mathrm{OPT}} \geqslant t + d(s,d) + d(d,o)$. Since either $d(s,o)$ or $d(d,o)$ is greater than $d(p,o)$ the triangle inequality gives $Z^{\mathrm{OPT}} \geqslant t + d(p,o)$. Hence, $Z^{\mathrm{TIR}} \leqslant 2Z^{\mathrm{OPT}}$.

Case 2b: Consider the set Q of rides that has been ignored temporarily by TIR. Let $\langle s_1,d_1\rangle$ be the ride in Q that is served first in the optimal off-line route, and let t_1 be the time at which this ride was presented. Moreover, let T_Q be an optimal route for the rides in Q. Then certainly $Z^{\mathrm{OPT}} \geqslant t_1 + |T_Q| - d(o,s_1)$. At t_1 TIR follows an optimal route to serve all yet unserved rides not in Q. It must have traveled already more than $d(o,s_1)$ on that route, since at the moment the ride $\langle s_1,d_1\rangle$ was presented it was in a position more remote from o. After finishing this route TIR makes an optimal route for serving the rides in Q. Therefore, $Z^{\mathrm{TIR}} \leqslant t_1 + |T| - d(s_1,o) + |T_Q| \leqslant 2Z^{\mathrm{OPT}}$. $\square$

As we noticed in the theorem the competitive ratio of TIR is tight, since it matches the lower bound from Theorem 2.1. Still, we give one example that shows asymptotic tightness. We learn from it that tightness of the algorithm occurs already on the real line as metric space and moreover that the worst-case occurs in a situation of coinciding source and destination for each ride, i.e., in an instance of the on-line TSP. The latter observation is not too surprising since rides of positive length give extra restrictions on the shape of the tour, also of the off-line optimal tour, whereas in the case of infinite capacity the server cannot be trapped during a ride that he is urged to finish before he can do anything else.

The worst-case example is defined on the line with 0 as the origin. At time 0 there is a request $\langle 1,1\rangle$. TIR would serve it immediately and therefore arriving in 1 at time 1. From there he proceeds back to 0. At time $1+\varepsilon$, a new request is presented in

$\langle 1, 1\rangle$. TIR will find himself in Case 2(a), and thus, returns to the origin 0, from where at time 2 he starts to serve the new ride, and therefore finishing at time 4. Obviously, an optimal service schedule would have finished at time $2 + \varepsilon$. Thus, the competitive ratio can be arbitrarily close to 2.

We notice that the lower bound in Theorem 2.1 does not necessarily hold for the real line and therefore a better algorithm for that particular metric space might exist.

It might be interesting to study the problem in which preemption of rides is allowed, i.e., a ride may be interrupted at any point and resumed at that point later again at no extra cost. The off-line version of this problem has been studied in [10].

3. Competitiveness of the latency problem

We assume the same setting as described precisely in the previous section. Again a single server starting in the origin is to serve requests for rides. Instead of minimizing the makespan we consider here the problem of minimizing *latency*, i.e., the average completion time or the sum of the completion times of the rides. The completion time of a ride is the time at which the ride from source to destination is completed. The restriction that we had in the previous section, that the server is obliged to return to the origin, is dropped here since it will not be accounted for in the objective function. The single server has a certain capacity being the number of rides he can be serving simultaneously.

A lower bound on the competitive ratio of any deterministic algorithm for this problem for any capacity will be derived using an adversary that gives a sequence of ride requests playing against an algorithm for the on-line problem.

Theorem 3.1. *No deterministic algorithm for the single-server on-line dial-a-ride problem on any metric space in the class $\mathscr{M}$ minimizing the latency can have a competitive ratio less than $1 + \sqrt{2}$, independent of the capacity of the server.*

Proof. The lower bound will be proved by considering the problem defined on the real line as the underlying metric space with 0 as the origin. Think of an adversary that at time $t = 0$ presents a request for the ride $\langle -1, -1\rangle$. Any algorithm should at some time, say at time x, serve that request. If $x > 1 + \sqrt{2}$ the adversary will not present any further requests and the algorithm is more than $1 + \sqrt{2}$-competitive, since the adversary will complete this single ride at time 1.

If $x \leqslant 1 + \sqrt{2}$ then at time $t = x$ the adversary presents a large number n of equal rides $\langle x, x\rangle$. In that case, the adversary will first complete the n rides before serving the ride in $\langle -1, -1\rangle$. Its sum of completion times is then $nx + 2x + 1$. The algorithm is at time x in -1, serves the ride there, and hence cannot complete the n rides before time $2x + 1$. Thus, its sum of completion times is bounded from below by $x + n(2x + 1)$. Hence, a lower bound on the competitive ratio of the algorithm is $(n(2x + 1) + x)/(nx + 2x + 1)$. Taking n arbitrarily large, this ratio gets arbitrarily close to $(2x + 1)/x$, which is a monotonically decreasing function of x on $[0,\infty]$, and therefore the best algorithm that does not arrive in -1 after time $1+\sqrt{2}$ will arrive there exactly at that time, yielding a competitive ratio of

$$\frac{2+2\sqrt{2}+1}{1+\sqrt{2}} = 1+\sqrt{2}.$$

We notice that since the source and destination of each ride in the adversarial sequence coincide, the capacity of the server does not play any role. Therefore, the above lower bound holds for any capacity of the server. □

In case the capacity of the server is 1, we can obtain a higher lower bound of 3.

Theorem 3.2. *No deterministic algorithm for the on-line dial-a-ride problem on any metric space in the class $\mathcal{M}$ with a single server having unit capacity and minimizing latency can have a competitive ratio of less than* 3.

Proof. At time $t=0$, the adversary presents a request for the ride $\langle 0,-1\rangle$. If an algorithm starts this ride not before time 2, there will be no further requests and the algorithm has competitive ratio at least 3.

If an algorithm starts the ride at a time $0<x<2$, then at time x the adversary presents n requests for the ride $\langle x,x\rangle$. The algorithm will first finish the first ride at time $x+1$, after which the n rides are completed at time $x+2+x\geqslant 3x$. The adversary first completes the n rides presented at time x. Thus, the competitive ratio is at least $(n3x+x+1)/(nx+2x+1)$, which can be made arbitrarily close to 3, by choosing n large enough.

Finally, if an algorithm starts the ride at time $x=0$, n requests for the ride $\langle 1,1\rangle$ are presented. The algorithm will not complete the n rides before time $1+2=3$, whereas the adversary will complete the n rides at time 1. This yields a competitive ratio of $(3n+1)/(n+3)$, which is again arbitrarily close to 3 by choosing n large enough. □

From the adversarial sequence in the above proof we deduce that any server with unit capacity that starts a ride as soon as one is presented, i.e., does not wait, will not have a constant competitiveness.

Lemma 3.1. *For the on-line dial-a-ride problem with a single server with capacity 1 and minimizing latency, any algorithm that starts serving a request immediately whenever there is any one unserved will be $\Omega(n)$-competitive, with n denoting the total number of requests.*

Proof. The proof is a simple adaptation of the adversarial sequence in the proof of the previous theorem. At time 0 the request $\langle 0,-1\rangle$ is given, and at time ε follow n requests $\langle 0,0\rangle$. The adversary will wait ε and hence will reach a latency of $(n+1)\varepsilon+1$. An algorithm that does not wait will have to finish the first ride before returning to 0 to serve the n later rides hence yielding a latency of $1 + 2n$. Letting ε tend to 0 gives the desired lower bound. □

For the single-server problem with any capacity greater than 1 similar conclusions can be drawn based on the adversarial sequence in the proof of Theorem 3.1. In particular, any algorithm that does not allow to wait will have a competitive ratio of at least 3.

We will now show a competitive algorithm for the capacity ∞ case on the line. The same algorithm is 9-competitive for the case in which sources and destinations coincide, that is the on-line version of the so-called TRP. The algorithm is an adaptation of the 9-competitive algorithm proposed by Baeza-Yates et al. in [5] for the problem of searching a point on a line. The same algorithm has been proposed in [6] as an approximation algorithm for the off-line TRP on the line. Our algorithm, that we call BCR (for Blindly Construct the Route) works as follows: Let $\sigma = \min\{t, |u|\}$, where u is the position of the starting point nearest to the origin of the rides that are presented at time 0, if any, and $t > 0$ is the first time strictly after 0 at which a request for a ride is presented.

At time 0 the server starts moving to the right until it arrives to point σ, then back to 0 and to the left till -2σ, then to 4σ and so on. In general, phase k of the algorithm consists of moving from 0 to $(-2)^k\sigma$ and back to 0, $k = 0, 1, 2, \ldots$. At every point where there is a request the server picks it up and delivers it at its destination the first time he visits it. We will prove that this algorithm is 9-competitive for the case in which sources and destinations coincide (on-line TRP) and 15-competitive for the latency problem with capacity ∞.

Theorem 3.3. *BCR has competitive ratio* 9 *for the on-line Traveling Repairman Problem on the line.*

Proof. Without loss of generality we may assume that $\sigma = 1$. We will prove the stronger result that any single request is served not later than 9 times the minimum time it could have been served. Let us study the request s presented at time t. A first observation is that the optimal completion time of this request is

$$C_s^{\mathrm{OPT}} \geqslant \max\{|s|, t\} \geqslant \sigma = 1. \tag{2}$$

Moreover, from [5] we know that if s is served the first time it is visited then its completion time achieved by BCR will be at most nine times the optimal completion time. It remains to show that the same bound holds if this is not true.

Assume that s is served during phase k of BCR. For $k < 2$ the theorem is obviously true since phase 1 finishes at time $2(2^0 + 2^1) = 6 \leqslant 6C_s^{\mathrm{OPT}}$. Thus, we assume that $k \geqslant 2$. We distinguish between two cases.

1. s is served while the server is moving from 0 to $(-2)^k$, i.e., in the first half of phase k. Since it is not the first visit of BCR at s, we must have that

   $$|s| \leqslant 2^{k-2},$$

and also

$$t \geqslant 2\sum_{i=0}^{k-3} 2^i + 2^{k-2} = 2(2^{k-2} - 1) + 2^{k-2} = 3(2^{k-2}) - 2. \tag{3}$$

For the completion time of the request by BCR we have

$$\begin{aligned} C_s^{\mathrm{BCR}} &= 2\sum_{i=0}^{k-1} 2^i + |s| \leqslant 2(2^k - 1) + 2^{k-2} = 9(2^{k-2}) - 2 \\ &\leqslant t + 6(2^{k-2}) \leqslant 7C_s^{\mathrm{OPT}}, \end{aligned}$$

where the last but one inequality follows from (3) and the last inequality from (2) together with the fact that from (3) we have $t \geqslant 2^{k-2}$ for $k \geqslant 2$.

2. s is served while the server is moving from $(-2)^k$ to 0. But this means that the request has been presented after the server started phase k, and therefore,

$$t \geqslant 2\sum_{i=0}^{k-1} 2^i = 2(2^k) - 2. \tag{4}$$

In this case

$$C_s^{\mathrm{BCR}} \leqslant 2\sum_{i=0}^{k} 2^i = 2(2^{k+1} - 1) = 4(2^k) - 2 \leqslant t + 2(2^k) \leqslant 3C_s^{\mathrm{OPT}},$$

where the last but one inequality follows from (4) and the last inequality from (2) and the fact that from (4) we have $t \geqslant 2^k$ for $k \geqslant 2$.

Since the above shows that for any job its completion time achieved by BCR is at most nine times its earliest possible completion time, the theorem follows. The tightness of this bound follows directly from [5]. □

Theorem 3.4. *BCR has a competitive ratio of* 15 *for the on-line dial-a-ride problem on the line with a single server having infinite capacity and minimizing the latency.*

Proof. The proof is along the same lines as that of the previous theorem. Again, without loss of generality we assume that $\sigma = 1$, and study the behaviour of BCR on some arbitrary request $\langle s, d\rangle$ presented at time t. Obviously, the optimal completion time of this ride is

$$C_{s,d}^{\mathrm{OPT}} \geqslant \max\{|s|, t\} + d(s,d) \geqslant \sigma + d(s,d) \geqslant 1. \tag{5}$$

From [5], we know that if the ride is completed in d the first time that d is visited then its completion time achieved by BCR will be at most 9 times the optimal completion time.

In studying the remaining situation, assume that the ride has started in s during phase k of BCR. In case $k = 0$, the theorem is obviously true since the ride will be completed at latest in phase 2, in which case $d \leqslant 1$, and therefore the completion time is at most $2+4+1=7 \leqslant 7C^{\mathrm{OPT}}_{s,d}$. If $k=1$, the ride will be completed at the latest in phase 3 at time $2+4+8+|d|=14+|d| \leqslant 15C^{\mathrm{OPT}}_{s,d}$. Thus, we assume from now on that $k \geqslant 2$. We distinguish between two cases.

1. $\langle s,d \rangle$ is started in s while the server is moving from 0 to $(-2)^k$. For the completion time of the request by BCR we have

$$\begin{aligned} C^{\mathrm{BCR}}_{s,d} &\leqslant 2\sum_{i=0}^{k-1} 2^i + |s| + 2(2^k - |s|) + d(s,d) \\ &= 2(2^k-1) + 2(2^k) + d(s,d) - |s| \\ &= 16(2^{k-2}) + d(s,d) - |s| - 2. \end{aligned} \tag{6}$$

   We consider two further subcases this time, the first of which is the crucial one.

   - When starting the ride in s, it is the first visit of BCR at s, in which case $|s| \geqslant 2^{k-2}$. In this case we can bound the BCR completion time of the ride in (6) by

   $$C^{\mathrm{BCR}}_{s,d} \leqslant 15|s| + d(s,d) - 2 \leqslant 15C^{\mathrm{OPT}}_{s,d},$$

     using (5).
   - When starting the ride in s, it is not the first visit of BCR at s, in which case $t \geqslant 2\sum_{i=0}^{k-3} 2^i + 2^{k-2} = 3(2^{k-2}) - 2$. In this case we bound the BCR completion time of the ride in (6) by

   $$C^{\mathrm{BCR}}_{s,d} \leqslant 16(2^{k-2}) + d(s,d) - 2 \leqslant 13(2^{k-2}) + t + d(s,d) \leqslant 14C^{\mathrm{OPT}}_{s,d},$$

   using (5) and the bound on t for this case, which also implies that $t \geqslant 2^{k-2}$ for $k \geqslant 2$.
2. $\langle s,d \rangle$ is started in s while the server is moving from $(-2)^k$ to 0. This means that the request has been presented after the server started phase k, and therefore $t \geqslant 2\sum_{i=0}^{k-1} 2^i = 2(2^k) - 2$. In this case

$$C^{\mathrm{BCR}}_{s,d} \leqslant 2\sum_{i=0}^{k+1} 2^i + 2^k = 2(2^{k+2}-1) + 2^k = 9(2^k) - 2 \leqslant 7(2^k) + t \leqslant 8C^{\mathrm{OPT}}_{s,d},$$

   where the last inequality follows from (5) and the fact that $t \geqslant 2^k$.

Since the above shows that for any ride its completion time achieved by BCR is at most 15 times its earliest possible completion time, the theorem follows.

That the ratio is tight is seen from the following request sequence. At time 0 one request is presented for ride $\langle 1,1 \rangle$ and one for ride $\langle 2^{k-2}+2\varepsilon, 2^{k-2}+\varepsilon \rangle$. The first ride is completed at time $C^{\mathrm{BCR}}_1 = 1$ and the completion time C^{BCR}_2 of the second ride is exactly

$$C^{\mathrm{BCR}}_2 = 16(2^{k-2}) + \varepsilon - 2^{k-2} - 2\varepsilon - 2 = 15(2^{k-2}) - 2 - \varepsilon.$$

Thus, $Z^{\mathrm{BCR}} = C^{\mathrm{BCR}}_1 + C^{\mathrm{BCR}}_2 = 15(2^{k-2}) - 1 - \varepsilon$.

Obviously, for the optimal solution we have $C_1^{\mathrm{OPT}} = 1$ and $C_2^{\mathrm{OPT}} = 2^{k-2} + 3\varepsilon$, and $Z^{\mathrm{OPT}} = 2^{k-2} + 1 + 3\varepsilon$. The ratio tends to 15 if k tends to ∞. $\square$

4. Epilogue

We emphasize that the present paper contains first results in a field of natural on-line problems that has so far been virtually unexplored. On-line dial-a-ride problems are occurring in a wide variety of practical settings, and cover not only physical rides by transportation means. As an abstraction almost all (machine) scheduling and routing problems can be translated as special versions of dial-a-ride.

That the field raises theoretical challenges is well exemplified by the problems with a criterion of minimizing latency. The constant competitive algorithm that we presented does not take into account any information about the locations of the rides, but just moves blindly from left to right and vice versa. Indeed, there is a rather big gap between its competitive ratio and the lower bound. An example shows that the obvious and intuitively not so bad greedy algorithm has an infinite competitive ratio. Copying of the results for the related on-line single-machine scheduling problem [18, 21] is not straightforward, since there are no easy lower bounds on completion times of rides, e.g., given by the preemptive version of the problem.

Considering multiple servers would be a stimulating extension to our framework, both from a theoretical and practical point of view. Besides, it is interesting to introduce more realistic restrictions to the model, as restrictions on the minimum and maximum length of rides (relative to the speed of the server), maximum individual waiting times, and others that may arise in a practical setting.

Acknowledgements

We are indebted to Marcelo Mydlarz for discussions and suggestions that lead to some of the results for the latency problem. We also wish to thank Sven Krumke for pointing out a flaw in a previous manuscript of the paper.

References

[1] N. Ascheuer, M. Groetschel, N. Kamin, J. Rambau, Combinatorial online optimization in practice, preprint SC 98-07, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1998.
[2] N. Ascheuer, S.O. Krumke, J.Rambau, Competitive scheduling of elevators, preprint SC 93-34, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1998.
[3] M.J. Atallah, S.R. Kosaraju, Efficient solutions to some transportation problems with application to minimizing robot arm travel, SIAM J. Computing 17 (1988) 849–869.
[4] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, M. Talamo, Algorithms for the on-line traveling salesman, Algorithmica, to appear.
[5] R. Baeza-Yates, J. Culberson, G. Rawlins, Searching in the plane, Inform. and Comput. 106 (2) (1993) 234–252.

[6] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, M. Sudan, The minimum latency problem, Proc. 26th Annual ACM Symp. on Theory of Computing, 1994, pp. 163–171.

[7] L.D. Bodin, B.L. Golden, A. Assad, M.O. Ball, Routing and scheduling of vehicles and crews: the state of the art, Comput. Oper. Res. 10 (1983) 63–211.

[8] M. Charikar, B. Raghavachari, The finite capacity dial-a-ride problem, Proc. 39th Ann. Symp. on Foundations of Computer Science, 1998 pp. 458–467.

[9] E. Feuerstein, M. Seleson, A. Strejilevich de Loma, On-line multi-threaded dial a ride problems, manuscript, 1999.

[10] G.N. Frederickson, D.J. Guan, Preemptive ensemble motion planning on a tree, SIAM J. Comput. 21 (1992) 1130–1152.

[11] G.N. Frederickson, D.J. Guan, Nonpreemptive ensemble motion planning, J. Algorithms 15 (1993) 29–60.

[12] G.N. Frederickson, M.S. Hecht, C.E. Kim, Approximation algorithms for some routing problems, SIAM J. Comput. 7 (1978) 178–193.

[13] D.J. Guan, Routing a vehicle of capacity greater than one, Discrete Appl. Math. 81 (1998) 41–57.

[14] R.M. Karp, Two combinatorial problems associated with external sorting, Combinatorial Algorithms, Courant Computer Science Symp., Algorithmics Press, New York, 1972, pp. 17–29.

[15] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Sequencing and scheduling: algorithms and complexity, in: Handbooks in Operations Research and Management Science, Vol. 4, Elsevier Science Publishers, Amsterdam, 1993 (Chapter 9).

[16] J.K. Lenstra, W.E. de Paepe, J. Sgall, R.A. Sitters, L. Stougie, Classification of dial-a-ride problems, Manuscript.

[17] W.E. de Paepe, Computer-aided complexity classification of dial-a-ride problems, M.Sc. Thesis, University of Amsterdam, 1998.

[18] C.A. Phillips, C. Stein, J. Wein, Scheduling jobs that arrive over time, in: S.G. Akl, F. Dehne, J.-R. Sack, N. Santoro (Eds.), Proc. 4th WADS: Workshop on Algorithms and Data Structures, Kingston, Canada, August 16–18, 1995, Lecture Notes in Computer Science, vol. 955, Springer, Berlin, 1995, pp. 86–97.

[19] M.W.P. Savelsbergh, M. Sol, The general pickup and delivery problem, Transportation Sci. 29 (1995) 17–29.

[20] D.B. Shmoys, J. Wein, D.P. Williamson, Scheduling parallel machines on-line, SIAM J. Comput. 24 (1995) 1313–1331.

[21] A. Vestjens, On-line machine scheduling, Ph.D. Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, 1997.