Eindhoven University
of Technology

**Department of Mathematics
and Computing Sciences**

Memorandum COSOR 98-06

On-line single server dial-a-ride problems

E. Feuerstein, L. Stougie

Eindhoven, March 1998
The Netherlands

On-line single server dial-a-ride problems

Esteban Feuerstein [*] Leen Stougie[†]

March 17, 1998

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 travelling. This models e.g. the taxi problem, or, if the server has capacity more than 1 a minibus or courier service problem. For two versions of this problem, one in which the server has infinite capacity and the other in which the server has capacity 1, both having as objective minimization of the time the last destination is served, we will design algorithms that have competitive ratio's of 2. We also show that these are best possible, since no algorithm can have competitive ratio better than 2 for these problems.

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 servers with capacity 1.

Keywords: Dial-a-ride, on-line optimization, competitive analysis.

---

[*] Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, and Instituto de Ciencias, Universidad de General Sarmiento, Argentina. 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". e-mail: efeuerst@dc.uba.ar

[†] Combinatorial Optimization Group, Faculty of Mathematics, Technical University Eindhoven, P.O. Box 513, 5600MB Eindhoven, The Netherlands. Supported by the Human Capital Mobility Network DONET of the European Community. e-mail: leen@win.tue.nl

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 start point of the ride, and a *destination*, which is the endpoint 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 another paper [3] we have proposed a classification for dial-a-ride problems similar to that developed for scheduling problems in [7].

In this paper we study *on-line* dial-a-ride problems. It is almost in the name that the on-line setting is a very natural one for these problems. Requests for rides come in over time, while the server(s) are en route 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.

Some studies on dial-a-ride problems have appeared in the literature [4], [2], [5], [6], be it under various names, like stacker crane problems, and ensemble motion planning problems. They all concern the off-line problem. In these papers it is proved that the general problem with one server having capacity 1 and minimizing the time to serve all rides is NP-hard. On the line as underlying metric space the problem is solvable in polynomial time. On a tree the problem is already NP-hard.

As far as we know no results on the on-line dial-a-ride problem have appeared in the literature. Closely related to the problems considered here is the work on the on-line Travelling Salesman Problem [1]. The TSP can be seen as a special version of the dial-a-ride problem in which for each request source and destination coincide.

We will design algorithms for the problems with a single server on a metric space that satisfies some conditions exposed at the end of this section. The objective is to minimize the time it takes to serve all the rides and return in the origin. We call this the *makespan* of the service, in analogy to the makespan in scheduling problems. Two cases are considered. In the first case the server has capacity 1. In the second case it has infinite capacity.

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 show in Section 2 that our algorithms have competitive ratio 2, and that this is best possible by providing a lower bound of 2 on the competitive ratio of any algorithm for the on-line problems.

After that in Section 3 we make a start with the study of the dial-a-ride problem with the objective of minimizing the average completion time of the requests, also called the *latency*. Even the complexity of the off-line version of this problem is unknown. The only 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 [3]. We conjecture that the single server capacity 1 problem is already NP-hard.

We will prove a lower bound of $1 + \sqrt{2}$ on the competitive ratio of any 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. So far no constant competitive algorithms are known.

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.

We finish this section by defining a class $\mathcal{M}$ of metric spaces that will be considered in the rest of the paper. Every metric space in $\mathcal{M}$ 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. $\mathcal{M}$ 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 $\mathcal{M}$ 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 remain there or move 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 $< s_j, d_j >$, with s_j the source and d_j the destination being points in some underlying metric space belonging to the class $\mathcal{M}$ 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 two variations. In the first one the capacity of the server is infinite, which means that he can collect any number of sources before getting to their destinations. In the second one the server has capacity 1, which means that as soon as he has started a ride he must finish this ride before being able to start any other ride. In both 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 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 $\mathcal{M}$, minimizing makespan, any 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 Travelling Salesman Problem, [1, 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. ☐

For the two on-line problems we propose the following algorithms that determine at each time t the behaviour of the server. They are based on the algorithm that was proposed for the on-line TSP in [1], which has been called PAH there, and in fact have the same competitive ratio. We call the algorithms TIR∞ (for Temporarily Ignore Requests) and TIR1, for the infinite capacity and the unit capacity problem, respectively.

We denote the position of the TIR∞ and the TIR1 algorithm at time t by p. First follows a description of the algorithm TIR∞.

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.

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:

2a. 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.

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

Theorem 2.2 *TIR∞ is has competitive ratio 2 for the on-line dial-a-ride problem with a single server having infinite capacity and minimizing total time used for serving all rides and returning in the origin. Therefore this algorithm is best possible.*

PROOF. Let t be the time the last request is presented and that request is the ride $< s, d >$. Let p be the position of TIR∞ at time t. Moreover, let T be an optimal route that makes all rides requested, and let $|T|$ denote its length. 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 $Z^{TIR\infty} \leq t + |T|$. Obviously, $|T|$ is a lower bound on the optimal off-line solution value: $Z^{OPT} \geq |T|$. Another trivial lower bound is $Z^{OPT} \geq t$. Thus, $Z^{TIR\infty} \leq 2Z^{OPT}$.

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

- Case 2b. Consider the set Q of rides that has been postponed by TIR∞. Let $< s_1, d_1 >$ 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 started at s_1 in the optimal off-line solution. Moreover, let T_Q be an optimal route for the rides in Q. Then certainly $Z^{OPT} \geq t_1 + |T_Q| - d(o, s_1)$. At t_1 TIR∞ follows an optimal route to serve all yet unserved rides not in Q. It must have travelled already more than $d(o, s_1)$ on that route, since at the moment the ride $< s_1, d_1 >$ 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, $Z^{TIR\infty} \leq t_1 + |T| - d(s_1, o) + |T_Q| \leq 2Z^{OPT}$.

□

For the problem in which the capacity of the server is 1 the above algorithm has to be adapted to deal with the situation that the server is serving a ride when the new request is presented. In that case it should finish its current ride first before doing anything else. This leads to algorithm TIR1.

1. If the server is in the origin o and is empty, i.e., is not serving a ride, then the optimal tour through all yet unserved requests is computed and followed.
2. If at time t a new request is presented with source s and destination d, then
   - 2a. If $d(s,o) > d(p,o)$ or $d(d,o) > d(p,o)$, and the server is not serving any ride at time t, then the server goes back to the origin (following the shortest path from p), where it appears in a Case 1 situation.
   - 2b. If $d(s,o) > d(p,o)$ or $d(d,o) > d(p,o)$, and the server is serving the ride $< x, y >$ at time t, then the server proceeds to y and computes in y an optimal path from y through all unserved requests to o.
   - 2c. If both $d(s,o) \leq d(p,o)$ and $d(d,o) \leq d(p,o)$ then the server ignores it until a new tour through unserved rides is computed, either after a Case 1 occurrence or the arrival at a destination of a ride after a Case 2b occurrence.

Theorem 2.3 *TIR1 has competitive ratio 2 for the on-line dial-a-ride problem with a single server having capacity 1 and minimizing the total time used to serve all rides and returning in the origin. Therefore it is best possible.*

PROOF. The proof is similar to that of the previous theorem. Again we assume that t is the time the last request for a ride is presented and the ride is $< s, d >$. The proof of Case 1 is equivalent to that in the previous theorem. The same holds for the proof of Case 2a.

In Case 2b the length of the path from y to o is bounded by $|T| - d(x,y) + d(x,o)$, with T again the shortest tour for all requests starting and finishing in o. Therefore,

$$\begin{aligned} Z^{TIR1} &\leq t + d(p,y) + |T| - d(x,y) + d(x,o) \\ &\leq t + |T| + d(p,y) - d(x,y) + d(x,p) + d(p,o) \\ &= t + |T| + d(p,o). \end{aligned}$$

For the optimal solution we have $Z^{OPT} \geq |T|$ and $Z^{OPT} \geq t + d(s,d) + d(d,o) \geq t + d(p,o)$. Thus, indeed $Z^{TIR1}/Z^{OPT} \leq 2$.

If in Case 2c the set Q of temporarily ignored requests is emptied at the moment the server arrives at o, the proof of Case 2b in the previous theorem still holds.

Suppose now that Q is emptied upon arrival at a destination y of a ride $< x, y >$. The final request $< s, d >$ has not caused this Case 2b situation.

Thus, at time t the server was already finishing the ride $< x, y >$ after a Case 2b occurrence during this ride at an earlier time, t'. Since, the request has arrived before arrival of the server at y, the new request $< s, d >$ will just be included in the optimal path from y to o serving all yet unserved requests. 2-competitiveness follows directly from the above Case 2b analysis with t' replacing t.

□

As we noticed in the two theorems the competitive ratio's of TIR∞ and TIR1 are tight, since they match the lower bound from Theorem 2.1. Still, we give one example that shows asymptotic tightness of both algorithms. We learn from it that tightness of the algorithms 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 a ride of length 0 gives much more flexibility to the adversary player. However, we have not been able so far to prove a general statement of this form.

The worst-case example is defined on the line with 0 as the origin. At time 0 there is a request $< 1, 1 >$. Both TIR∞ and TIR1 would serve it immediately and therefore arriving in 1 at time 1. From there they proceed back to 0. At time $1 + \epsilon$ a new request is presented in $< 1, 1 >$. Both algorithms will find themselves in Case 2a, and thus, return to the origin 0, from where at time 2 they start to serve the new ride, and therefore finishing at time 4. Obviously, an optimal service schedule would have finished at time $2+\epsilon$. 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 [5].

3 Lower bounds for the latency problem

Instead of minimizing the makespan as in the previous section, 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 single server has capacity c, which may also be infinite.

A lower bound on the competitive ratio of any algorithm for this problem for any c 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 *For any c, no algorithm for the single server on-line dial-a-ride problem on any metric space minimizing the latency can have a competitive ratio less than $1 + \sqrt{2}$.*

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 $< -1, -1 >$. Any algorithm should at some time, x say, 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 $x \leq 1 + \sqrt{2}$ then at time $t = x$ the adversary presents n equal rides $< x, x >$. In that case the adversary will first complete the n rides before serving the ride in $< -1, -1 >$. Its sum of completion times is then $nx + x + 1$. The algorithm is at time x in -1, serves the ride there, and proceeds to x to complete the n rides. Thus, its sum of completion times is $x + n(2x + 1)$. Hence the competitive ratio of the algorithm is $\frac{n(2x+1)+x}{nx+x+1}$ Taking n arbitrarily large, this ratio gets arbitrarily close to $\frac{2x+1}{x}$, 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 c. □

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

Theorem 3.2 *Any algorithm for the single server dial-a-ride problem with unit capacity on any metric space minimizing latency has a competitive ratio of at least 3.*

PROOF. At time $t = 0$ the adversary presents a request for the ride $< 0, -1 >$. 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 $< x, x >$. The algorithm will first finish the first ride at time $x + 1$, after which the n rides are completed at time $x + 2 + x \geq 3x$. The adversary first completes the n rides presented at time x. Thus, the competitive ratio is at least $n3x/(nx + 2x + 1)$, 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 $< 1, 1 >$ 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.

□

So far any attempt to devise an algorithm with a constant competitive ratio for this problem failed.

From the adversarial sequence in the above proof we deduce that any algorithm that starts a ride as soon as possible after a fraction x has passed of the earliest possible completion time minus the release time of the ride, will be at least $\rho(x)$-competitive with $\rho(x) = \max\{x + 1, (2x + 2)/x\}$. Thus, e.g. any algorithm that takes as this fraction any $x \neq 2$ has competitive ratio strictly greater than 3.

The situation is even more dramatic in case $x = 0$, i.e., the server moves immediately if any ride is unserved.

Lemma 3.1 *Any algorithm for the single server problem with capacity 1 that moves immediately whenever there is any unserved ride will be $O(n)$ competitive.*

PROOF. The proof is a simple adaption of the adversarial sequence in the proof of the previous theorem. At time 0 the request $< 0, -1 >$ is given, and at time ϵ follow n requests $< 0, 0 >$. The adversary will wait ϵ and hence will reach a latency of $(n + 1)\epsilon + 1$. 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 infinite capacity 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.

4 Epilogue

We emphasize that the the present paper contains only 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 gives challenges is well exemplified by the problems with a criterion of minimizing latency. The first constant competitive algorithm is still to be found. 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 [8], [9] 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. Even the complexity of the off-line problem with or without preemption of rides is open yet.

For both the makespan and the latency problem it would be interesting to introduce more realistic aspects, like restrictions on the minimum and maximum length of rides (relative to the speed of the server). Moreover, introduction of multiple servers gives a very practical extension.

Acknowledgements

We are indebted to Marcelo Mydlarz for the lower bound of Theorem 3.2.

References

[1] G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, M. Talamo. *On line routing problems.* To appear in *Algorithmica.*

[2] 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.

[3] E. Feuerstein, L. Stougie. *A classification and complexity of dial-a-ride problems.* Manuscript.

[4] G.N. Frederickson, M.S. Hecht, C.E. Kim. Approximation algorithms for some routing problems. *SIAM J. Computing, 7*, 1978. 178–193.

[5] G.N. Frederickson, D.J. Guan. Preemptive ensemble motion planning on a tree. *SIAM J. Computing, 21*, 1992. 1130–1152.

[6] G.N. Frederickson, D.J. Guan. Nonpreemptive ensemble motion planning. *Journal of Algorithms, 15*, 1993. 29–60.

[7] 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, 4*, Chapter 9. Elsevier Science Publishers, Amsterdam. 1993.

[8] 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). *WADS: Workshop*

on Algorithms and Data Structures: Proceedings: 4th, Kingston, Canada, August 16-18, 1995. Lecture Notes in Computer Science 955, Spinger, Berlin, 1995. 86–97.

[9] A. Vestjens. *On-line machine scheduling.* PhD-thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven, 1997.