Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

DIETRICH HAUPTMEIER SVEN O. KRUMKE JÖRG RAMBAU

The Online Dial-a-Ride Problem under Reasonable Load

Preprint SC 99-08 (March 1999)

THE ONLINE DIAL-A-RIDE PROBLEM UNDER REASONABLE LOAD

DIETRICH HAUPTMEIER, SVEN O. KRUMKE, AND JÖRG RAMBAU

ABSTRACT. In this paper, we analyze algorithms for the online dial-a-ride problem with request sets that fulfill a certain worst-case restriction: roughly speaking, a set of requests for the online dial-a-ride problem is reasonable if the requests that come up in a sufficiently large time period can be served in a time period of at most the same length. This new notion is a stability criterion implying that the system is not overloaded.

The new concept is used to analyze the online dial-a-ride problem for the minimization of the maximal resp. average flow time. Under reasonable load it is possible to distinguish the performance of two particular algorithms for this problem, which seems to be impossible by means of classical competitive analysis.

1. INTRODUCTION

It is a standard assumption in mathematics, computer science, and operations research that problem data are *given*. However, many aspects of life are *online*. Decisions have to be made without knowing future events relevant for the current choice. Online problems, such as vehicle routing and control, management of call centers, paging and caching in computer systems, foreign exchange and stock trading, had been around for a long time, but no theoretical framework existed for the analysis of online problems and algorithms.

Meanwhile competitive analysis has become the standard tool to analyse online-algorithms [4, 6]. Often the online algorithm is supposed to serve the requests one at a time, where a next request becomes known when the current request has been served. However, in cases where the requests arrive at certain points in time this model is not sufficient. In [3, 5] each request in the request sequence is time stamped by its release time. A similar approach was used in [1] to investigate the *online dial-a-ride problem*—ONLINEDARP for short— which is the example for the new concept in this paper.

In the problem ONLINEDARP objects are to be transported between the vertices of a given graph G. A request consists of the objects to be transported and the corresponding source and target vertex of the transportation request. The requests arrive online and must be handled by a server which commences and ends its work at a designated origin vertex and which moves along the paths in G. The server picks up and drops objects at their starts and destinations. It is assumed that neither the release time of the last request nor the number of requests is not known in advance.

A feasible solution to an instance of the ONLINEDARP is a schedule of moves (i.e., a sequence of consecutive moves along edges of the graph together with their starting times) on the edges of G so that every request is served and that no request is picked up before its release time. The goal of ONLINEDARP is to find a feasible solution with "minimal cost", where the notion of "cost" depends on the objective function used.

Recall that an online-algorithm A is called *c-competitive* if there exists a constant c such that for any request sequence σ:

$$C_{\mathsf{A}}(\sigma) \leq c \cdot C_{\mathrm{OPT}}(\sigma),$$

where $C_{\mathsf{X}}(\sigma)$ denotes the objective function value of the solution produced by algorithm X on input σ and OPTdenotes an optimal offline algorithm.

Competitive analysis of ONLINEDARP provided the following (see [1]):

Research supported by the German Science Foundation (grant 883/5-2).

- There are competitive algorithms (IGNORE and REPLAN) for the goal of minimizing the *total completion time* of the schedule.
- For the task of minimizing the *maximal waiting time* or the *maximal flow time* there can be no competitive algorithm. In particular, the algorithms IGNORE and REPLAN have an unbounded competitive ratio.
- It is open whether there is a competitive algorithm for minimizing the *average flow time*. The algorithms IGNORE and REPLAN are not competitive for this objective function.

It should be noted that the corresponding offline-problems (where all requests are known at the start of the algorithm) are NP-hard to solve for the objective functions of average or maximal flow time [11]. The offline problem of minimizing the total completion time is polynomially solvable on special graph classes but NP-hard in general [8, 2, 7, 10].

If we are considering a continuously operating system with continuously arriving requests (i.e., the request set may be infinite) then the total completion time is meaningless. Bottom-line: in this case, the existing positive results cannot be applied and the negative results tell us that we cannot hope for performance guarantees that may be relevant in practice. In particular, the two algorithms IGNORE and REPLAN cannot be distinguished by classical competitive analysis.

The point here is that we do not know any notion from the literature to describe what a particular set of requests should look like in order to allow for a continuously operating system. In queuing theory this is usually modelled by a stability assumption: the rate of incoming requests is at most the rate of requests served. To the best of our knowledge, so far there has been nothing similar in the existing theory of discrete online-algorithms. Since in many instances we have no exploitable information about the distributions of requests we want to develop a worst-case model rather than a stochastic model for stability of a continuously operating system.

Our idea is to introduce the notion of Δ_0-*reasonable* request sets. A set of requests is Δ_0-reasonable if—roughly speaking—requests released during a period of time $\Delta \geq \Delta_0$ can be served in time at most Δ. A set of requests R is *reasonable* if there exists a $\Delta_0 < \infty$ such that R is Δ_0-reasonable. That means, for non-reasonable request sequences we find arbitrarily large periods of time where requests are released faster than they can be served—even if the server has the optimal offline schedule. When a system has only to cope with reasonable request sets, we call this situation *reasonable load*. Section 3 is devoted to the exact mathematical setting of this idea.

The main result on the ONLINEDARP under Δ_0-reasonable load is the following:

Theorem 1.1. *For the* ONLINEDARP *under Δ_0-reasonable load, IGNORE yields a maximal and an average flow time of at most $2\Delta_0$, whereas the maximal and the average flow time of REPLAN are unbounded.*

We prove this result in Sections 4 and 5.

2. PRELIMINARIES

Let us first sketch the problem under consideration. We are given a graph $G = (V,E)$ with nodes V (stop positions) and edges E (transportation network) with non-negative weights (traversal times). Moreover, there is a special vertex $o \in V$ (the origin). Requests are triples $r = (t,a,b)$, where a is the start node of a transportation task, b its end node, and t its release time, which is—in this context—the time where r becomes known. A *transportation move* is a quadruple $m = (t,a,b,R)$, where a is the starting point and b the end point of an edge in E, and t the starting time of the move, while R is the set (possibly empty) of carried requests carried by the move. The *arrival time* of a move is the sum of its starting time and the weight of the edge (a,b) in G. An *open transportation schedule* is a sequence $(m_1,m_2,\dots)$ of transportation moves such that

- the first move starts in the origin o;
- the starting point of m_i is the end point of m_{i-1};
- the starting time of m_i carrying R is no earlier than the maximum of the arrival time of m_i and the release times of all requests in R.

If we additionally require the last move to end in the origin o then we get a *closed* schedule. In the sequel, we deal with closed schedules only.

An *online-algorithm* for ONLINEDARP has to move a server along edges in E so as to fulfill all released transportation tasks, while it does not know about requests that come up in the future. In order to plan the work of the server, the online-algorithm may maintain a preliminary (closed) transportation schedule for all known requests, according to which it moves the server. A posteriori, the moves of the server induce a complete transportation schedule, i.e., an offline-solution to the ONLINEDARP that may be compared to an optimal transportation schedule according to some objective function (competitive analysis). For a detailed set-up see [1].

We start with some useful notation. The goal is to avoid the special meaning for time 0 in the analysis of online algorithms with time stamped requests.

Definition 2.1. For a request $r = (t, a, b)$ we denote

$$t(r) := t, \qquad a(r) := a, \qquad b(r) := b.$$

The *time shift* of r by $\tau \in \mathbb{R}$ is the request

$$r + \tau := (t + \tau, a, b)$$

The *offline version* of r is the request

$$r^{\mathit{offline}} := (0, a, b).$$

Definition 2.2. Let R be a request set for ONLINEDARP. The *time shift* of R by τ is the request set

$$R + \tau := \{r + \tau \;:\; r \in R\}.$$

The *offline version* of R is the request set

$$R^{\mathit{offline}} := \left\{ r^{\mathit{offline}} \;:\; r \in R \right\}.$$

An important characteristic of a request set with respect to system load considerations is the time period in which it is released.

Definition 2.3. Let R be a finite request set for ONLINEDARP. The *release span* $\Delta(R)$ of R is defined as

$$\Delta(R) := \max_{r \in R} t(r) - \min_{r \in R} t(r).$$

The next definition describes a class of objectives that make sense if we discard the special meaning of time 0.

Definition 2.4. A cost function C for the ONLINEDARP is *translation invariant* if for any set of requests $R = r_1, r_2, \ldots$ and for all algorithms A we have

$$C_{\mathsf{A}}(R) = C_{\mathsf{A}}(R + \tau).$$

Example 2.5.
- The *completion time* or *makespan* of a request set R, i.e., the time when all requests from R are served, is not translation invariant. However, if we set all release times in a request set R to 0 then we get the translation invariant objective $C^{comp}(R^{\mathit{offline}})$. This is the *offline completion time* of R.
- Assume the first request in R is released at time t_1. Then the *flow time of* R is the translation invariant objective $C^{comp}(R - t_1)$.

- The *average flow time* and the *maximal flow time* of a request set R, i.e., the average resp. maximum time taken over all request $r \in R$ that r spends in the system, are translation invariant. We consider these objectives as especially important for a continuously operating system.

The situation now is as follows: for translation invariant objectives we find only negative results in the literature. Provably good algorithms exist only for the total completion time and the weighted sum of completion times. How can we make use of these algorithms in order to get performance guarantees for the translation invariant cost functions? We suggest a way of characterizing request sets which "make sense" in the translation invariant setting.

3. Reasonable Load

In a continuously operating system we wish to guarantee that work can be accomplished at least as fast as it is presented. In the following we propose a mathematical set-up which models this idea in a worst-case fashion. Since we are always working on finite subsets of the whole request set the request set itself may be infinite, resembling a continuously operating system.

We start by relating the release spans of finite subsets of a request set to the time we need to fulfill the requests.

Definition 3.1. Let R be a request set for the OnlineDarp. A weakly monotone function

$$f\colon \begin{cases} \mathbb{R} & \to & \mathbb{R}, \\ \Delta & \mapsto & f(\Delta); \end{cases}$$

is a *load bound* on R if for any $\Delta \in \mathbb{R}$ and any finite subset S of R with $\Delta(S) \leq \Delta$ we have

$$C_{\mathsf{OPT}}^{comp}(S^{offline}) \leq f(\Delta).$$

Remark 3.2. If the whole request set R is finite then there is always the trivial load bound given by the total completion time of R. For every load bound f we may set $f(0)$ to be the maximum completion time we need for a single request, and nothing better can be achieved.

The most stable situation would be a load bound equal to the identity on $\mathbb{R}$. In that case we would never get more work to do than we can accomplish. However, we cannot expect that the identity is a load bound for any of our problems because of the following observation: a request set consisting of one single request has a release span of 0 whereas in general it takes non-zero time to serve this request. In the following definition we introduce a parameter describing how far a request set is from being load-bounded by the identity.

Definition 3.3. A load bound f is Δ_0-*reasonable* for some $\Delta_0 \in \mathbb{R}$ if

$$f(\Delta) \leq \Delta \quad \text{for all } \Delta \geq \Delta_0$$

A request set R is Δ_0-*reasonable* if it has a Δ_0-reasonable load bound.

Remark 3.4. If Δ_0 is sufficiently small so that all request sets consisting of two or more requests have a release span larger than Δ_0 then the first-come-first-serve strategy is good enough to ensure that there are never more than two unserved requests in the system. Hence, the request set does not require scheduling the requests in order to provide for a stable system. (By "stable" we mean that the number of unserved requests in the system does not become arbitrarily large.)

In a sense, Δ_0 is a measure for the combinatorial difficulty of the request set R. Thus it is natural to ask for performance guarantees for algorithms in terms of this parameter. This is done for the algorithm IGNORE in the next section.

4. PERFORMANCE GUARANTEES FOR IGNORE

We are now in a position to prove the performance guarantees for minimizing the maximal resp. average flow time in the ONLINEDARP for algorithm IGNORE stated in Theorem 1.1. We start by recalling the algorithm IGNORE from [1]

Definition 4.1 (Algorithm IGNORE)**.** Algorithm IGNORE works with an internal buffer. It may assume the following states (initially it is IDLE):

IDLE: Wait for the next point in time when requests become available. Goto PLAN.
BUSY: While the current schedule is in work store the upcoming requests in a buffer ("ignore them"). Goto IDLE if the buffer is empty else goto PLAN.
PLAN: Produce a preliminary transportation schedule for all currently available requests R (taken from the buffer) minimizing $C^{comp}(R^{offline})$. Goto BUSY.

Let us consider the intervals in which IGNORE organizes its work in more detail. The algorithm IGNORE induces a dissection of the time axis $\mathbb{R}$ in the following way: Because maximal flow time is a translation invariant objective function, we can assume, w.l.o.g., that the first set of requests arrives at time 0. Let Δ_1 be the time period the server works on the first available set of requests. Moreover, for $i > 1$ let Δ_i be the time period the server is working on the requests that have been ignored during the last Δ_{i-1} time units. Then the time axis is split into the intervals

$$[0,\Delta_1],[\Delta_1,\Delta_1+\Delta_2],[\Delta_1+\Delta_2,\Delta_1+\Delta_2+\Delta_3],\dots$$

Let us denote these intervals by $I_1, I_2, I_3, \dots$. Moreover, let R_i be the set of those requests that come up in I_i. Clearly, the complete set of requests R is the union of all the R_i.

At the end of each interval I_i we solve an offline problem: all requests to be scheduled are already available. The work on the computed schedule starts immediately (at the beginning of interval I_{i+1}) and is done Δ_{i+1} time units later (at the end of interval I_{i+1}). On the other hand, the time we need to serve the schedule is equal to the optimal completion time of $R_i^{offline}$. In other words:

Lemma 4.2. *For all $i > 0$ we have*

$$\Delta_{i+1} = C^{comp}_{\mathsf{OPT}}(R_i^{offline}).$$

Let us now recall and prove the first statement of Theorem 1.1.

Theorem 4.3. *Let $\Delta_0 > 0$. For all Δ_0-reasonable request sets algorithm IGNORE yields a maximal flow time of no more than $2\Delta_0$.*

Proof. Let r be an arbitrary request in R_i, i.e., r is released in I_i. By construction, the schedule containing r is finished at the end of interval I_{i+1}, i.e., at most $\Delta_i + \Delta_{i+1}$ time units later than r was released. Thus, for all $i > 0$ we get that

$$C^{maxflow}_{\mathsf{IGNORE}}(R_i) \le \Delta_i + \Delta_{i+1}.$$

If we can show that $\Delta_i \le \Delta_0$ for all $i > 0$ then we are done. To this end, let $f : \mathbb{R} \to \mathbb{R}$ be a Δ_0-reasonable load bound for R. Then $C^{comp}_{\mathsf{OPT}}(R_i^{offline}) \le f(\Delta_i)$ because $\Delta(R_i) \le \Delta_i$.

There are two cases:

- If $\Delta_i \ge \Delta_0$ then $f(\Delta_i) \le \Delta_i$ because f is Δ_0-reasonable.
- If $\Delta_i < \Delta_0$ then $f(\Delta_i) \le f(\Delta_0)$ because f—as a load bound—is a weakly monotone function. Thus, in this case—again because f is Δ_0-reasonable—we get $f(\Delta_i) \le \Delta_0$.

Together with Lemma 4.2 we get for all $i > 0$

$$\Delta_{i+1} = C^{comp}_{\mathsf{OPT}}(R_i^{offline}) \le f(\Delta_i) \le \max\{\Delta_i, \Delta_0\}.$$

Because the release span of the requests served during the time period Δ_1 is 0 by construction of IGNORE we know that $\Delta_1 \le \max\{0,\Delta_0\} = \Delta_0$. It follows by induction on i that $\Delta_i \le \Delta_0$, and we are done. ☐

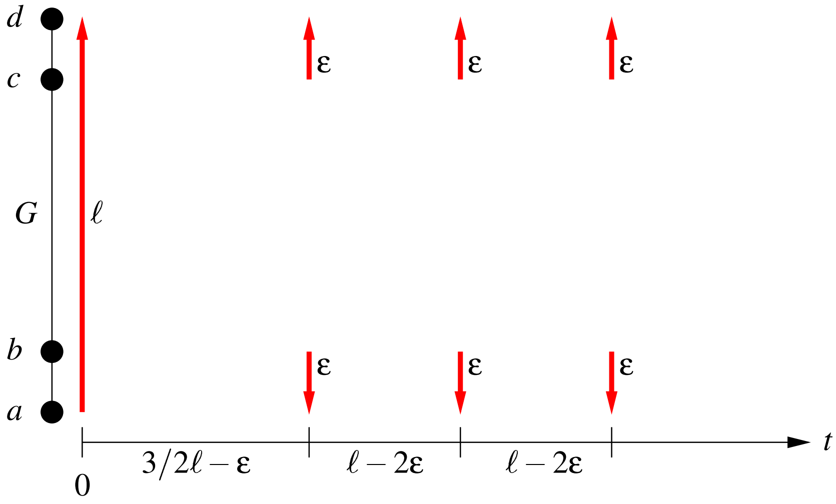

FIGURE 1. A sketch of a $(2\frac{2}{3}\ell)$-reasonable instance of ONLINEDARP ($\ell = 18\varepsilon$).

The statement of Theorem 1.1 concerning the average flow time of IGNORE follows from the fact that the average is never larger then the maximum.

Corollary 4.4. *Let $\Delta_0 > 0$. For all Δ_0-reasonable request sets algorithm IGNORE yields a average flow time no more than $2\Delta_0$.*

5. A DISASTROUS EXAMPLE FOR REPLAN

We first recall the strategy of algorithm REPLAN for the ONLINEDARP. Whenever a new request becomes available, REPLAN computes a preliminary transportation schedule from all available requests R, minimizing $C^{comp}(R^{offline})$. Then it moves the server according to that schedule until a new request arrives or the schedule is done. In the sequel, we provide an instance of ONLINEDARP and a Δ_0-reasonable request set R such that the maximal and the average flow time $C^{maxflow}_{\mathsf{REPLAN}}(R)$ is unbounded, thereby proving the remaining assertions of Theorem 1.1.

Theorem 5.1. *There is an instance of* ONLINEDARP *under reasonable load such that the maximal and the average flow time of REPLAN is unbounded.*

Proof. In Figure 1 there is a sketch of an instance for the ONLINEDARP. The graph G is a path on four nodes a,b,c,d; the length of the path is ℓ, the distances are $d(a,b) = d(c,d) = \varepsilon$, and hence $d(b,c) = \ell - 2\varepsilon$. At time 0 a request from a to d is issued; at time $3/2\ell - \varepsilon$, the remaining requests periodically come in pairs from b to a and form c to d, resp. The time distance between them is $\ell - 2\varepsilon$.

We show that for $\ell = 18\varepsilon$ the request set R indicated in the picture is $2\frac{2}{3}\ell$-reasonable. Indeed: it is easy to see that the first request from a to d does not influence reasonability. Consider an arbitrary set R_k of k adjacent pairs of requests from b to a resp. from c to d. Then the release span $\Delta(R_k)$ of R_k is

$$\Delta(R_k) = (k-1)(\ell - 2\varepsilon).$$

The offline version ${R_k}^{offline}$ of R_k can be served in time

$$C^{comp}_{\mathsf{OPT}}({R_k}^{offline}) = 2\ell + (k-1)\cdot 4\varepsilon.$$

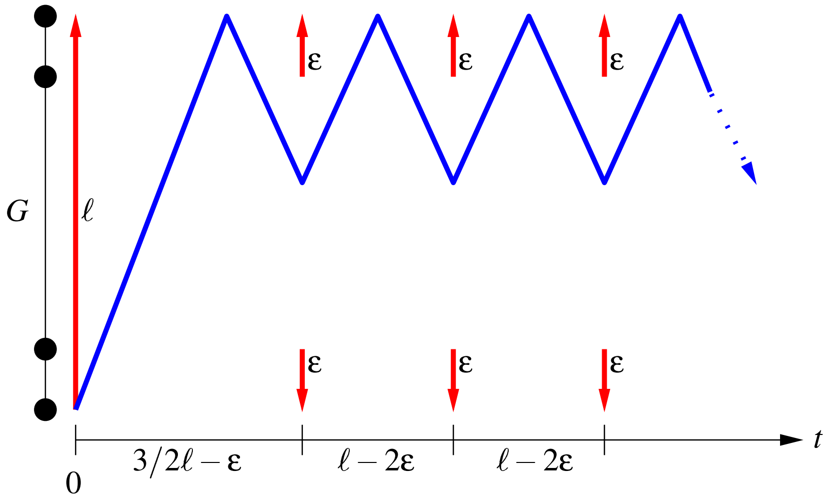

FIGURE 2. The track of the REPLAN-server. Because a new pair of requests is issued exactly when the server is still closer to the requests at the top all the requests at the bottom will be postponed in an optimal preliminary schedule. Thus, the server always returns to the top when a new pair of requests arrives.

In order to find the smallest paramter Δ_0 for which the request set R_k is Δ_0-reasonable we solve for the integer $k-1$ and get

$$k-1=\left\lceil \frac{2\ell}{\ell-6\varepsilon} \right\rceil = 3.$$

Hence, we can set Δ_0 to

$$\Delta_0 := C^{comp}_{\mathsf{OPT}}(R_4{}^{offline}) = 2\tfrac{2}{3}\ell.$$

Now we define

$$f : \begin{cases} \mathbb{R} & \to \quad \mathbb{R}, \\ \Delta & \mapsto \quad \begin{cases} \Delta_0 & \text{for } \Delta < \Delta_0, \\ \Delta & \text{otherwise.} \end{cases} \end{cases}$$

By construction, f is a load bound for R_4. Because the time gap after which a new pair of requests occurs is certainly larger than the additional time we need to serve it (offline), f is also a load bound for R. Thus, R is Δ_0-reasonable, as desired.

Now: how does REPLAN perform on this instance? In Figure 2 we see the track of the server following the preliminary schedules produced by REPLAN on the request set R.

The maximal flow time of REPLAN on this instance is realized by the flow time of the request $(3/2\ell-\varepsilon, b, a)$, which is unbounded.

Moreover, since all requests from b to a are postponed after serving all the requests from c to d we get that REPLAN produces an unbounded average flow time as well. □

In Figure 3 we show the track of the server under the control of the IGNORE-algorithm. After an initial inefficient phase the server ends up in a stable operating mode. This example also shows that the analysis of IGNORE in Section 4 is sharp.

6. CONCLUSION

We have introduced the mathematical notion Δ_0-reasonable describing the combinatorial difficulty of a possibly infinite request set for ONLINEDARP. For reasonable request

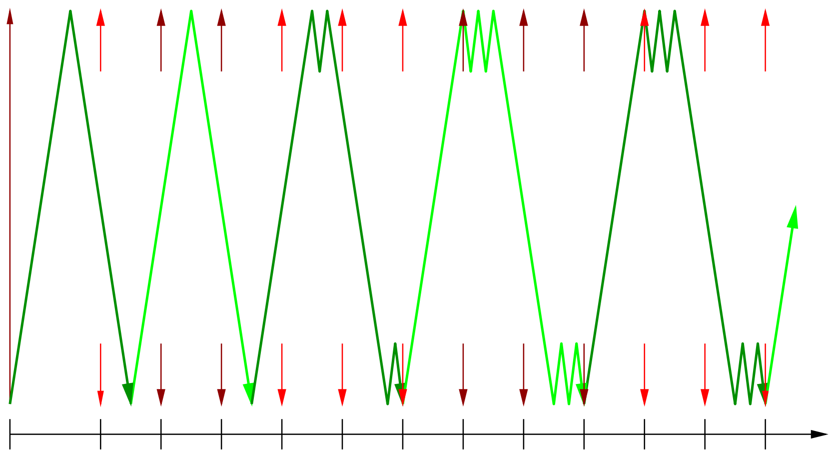

FIGURE 3. The track of the IGNORE-server.

sets we have given bounds on the maximal resp. average flow time of algorithm IGNORE for ONLINEDARP; in contrast to this, there are instances of ONLINEDARP where algorithm REPLAN yields an unbounded maximal and average flow time. One key property of our results is that they can be applied in continuously working systems. Computer simulations have meanwhile supported the theoretical results in the sense that algorithm IGNORE does not delay individual requests for an arbitraryly long period of time, whereas REPLAN has a tendency to do so [9].

While the notion of Δ_0-reasonable is applicable to minimizing maximal flow time, it would be of interest to investigate an average analogue in order to prove non-trivial bounds for the average flow times.

REFERENCES

1. Norbert Ascheuer, Sven O. Krumke, and Jörg Rambau, *The online transportation problem*, Preprint SC 98-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1998.
2. Mikhail J. Atallah and S. Rao Kosaraju, *Efficient solutions to some transportation problems with applications to minimizing robot arm travel*, SIAM Journal on Computing **17** (1988), 849–869.
3. Georgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and Maurizio Talamo, *Competitive algorithms for the traveling salesman*, Proceedings of the 4th Workshop on Algorithms and Data Structures (WADS'95), Lecture Notes in Computer Science, vol. 955, August 1995, pp. 206–217.
4. Allan Borodin and Ran El-Yaniv, *Online computation and competitive analysis*, Cambridge University Press, 1998.
5. Esteban Feuerstein and Leen Stougie, *On-line single server dial-a-ride problems*, Manuscript, submitted for publication, 1998.
6. Amos Fiat and Gerhard J. Woeginger (eds.), *Online algorithms: The state of the art*, Lecture Notes in Computer Science, vol. 1442, Springer, 1998.
7. Greg N. Frederickson and D. J. Guan, *Nonpreemptive ensemble motion planning on a tree*, Journal of Algorithms **15** (1993), 29–60.
8. Greg N. Frederickson, Matthew S. Hecht, and Chul Kim, *Approximation algorithms for some routing problems*, SIAM Journal on Computing **7** (1978), 178–193.
9. Martin Grötschel, Dietrich Hauptmeier, Sven O. Krumke, and Jörg Rambau, *Simulation studies for the online dial-a-ride-problem*, Preprint SC 99-09, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1999.
10. Dietrich Hauptmeier, Sven O. Krumke, Jörg Rambau, and Hans C. Wirth, *Euler is standing in line—dial-a-ride problems with fifo-precedence-contraints*, Preprint SC 99-06, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1999.
11. Hans Kellerer, Thomas Tautenhahn, and Gerhard Woeginger, *Approximability and nonapproximabiblity results for minimizing total flow time on a single machine*, Proceedings of the Symposium on the Theory of Computing, 1996.

{DIETRICH HAUPTMEIER, SVEN O. KRUMKE, JÖRG RAMBAU}, KONRAD-ZUSE-ZENTRUM FÜR INFORMATIONSTECHNIK BERLIN, TAKUSTR. 7, 14195 BERLIN, GERMANY

E-mail address: {hauptmeier,krumke,rambau}@zib.de