Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

BENJAMIN HILLER

Probabilistic Competitive Analysis of a Dial-a-Ride Problem on Trees under High Load

ZIB-Report 05-56 (December 2005)

Probabilistic Competitive Analysis of a Dial-a-Ride Problem on Trees under High Load

Benjamin Hiller*

Zuse Institute Berlin, Takustraße 7, D–14195 Berlin, Germany
hiller@zib.de

Abstract. In this paper we consider a simple variant of the Online Dial-a-Ride Problem from a probabilistic point of view. To this end, we look at a probabilistic version of this online Dial-a-Ride problem and introduce a probabilistic notion of the competitive ratio which states that an algorithm performs well on the vast majority of the instances. Our main result is that under the assumption of high load a certain online algorithm is probabilistically $(1+o(1))$-competitive if the underlying graph is a tree. This result can be extended to general graphs by using well-known approximation techniques at the expense of a distortion factor $\mathcal{O}(\log|V|)$.

1 Introduction

This paper deals with a certain elementary problem from a general class of transportation problems known as *Dial-a-Ride problems*. Dial-a-Ride problems arise in a wide range of applications. The elementary variant considered here was motivated by an application to a high rack warehouse.

We consider the following problem called Darp: There is one server which can serve at most one request at a time. It serves a request by traveling to its source location, picking up the object to transport and traveling to the destination location where the object is delivered. The server starts from a distinguished location (called *depot*) to which it has to return. Transportation is *non-preemptive*, i. e., once the service of a request has started there must not be an interruption until the request is finished at the destination location. We assume that we know all the distances between the locations. The objective is then to minimize the total travel distance, which is equivalent to minimizing total completion time (also known as *makespan*). Although the server (an elevator in the application) travels along a line it is convenient to model the transportation network by a special tree called caterpillar [1]. This is the reason for focusing on trees.

As is the case for many Dial-a-Ride applications there is an online aspect: Requests arrive over longer periods of time and should be served promptly. In this work we analyze the performance of an online algorithm on a random request sequence which tries to minimize total travel distance.

* Supported by the DFG research group "Algorithms, Structure, Randomness" (Grant number GR 883/9-3, GR 883/9-4).

There is a huge literature on Dial-a-Ride problems, both online and offline. Frederickson and Guan [2] showed that the Darp is NP-hard even on trees. Interestingly, the Darp can be solved in polynomial time on paths, but it is NP-hard on caterpillars [3, 4]. For the Darp on general graphs a $\frac{9}{5}$-approximation algorithm was presented by Frederickson et al. [5]. An improved algorithm for trees proposed by Frederickson and Guan [2] has a performance ratio of $\frac{5}{4}$. However, it has been observed that the solutions produced by another algorithm of Frederickson and Guan [2] called use-MST are optimal in many cases, whereas its approximation guarantee is only $\frac{4}{3}$. This motivated the probabilistic analysis of Coja-Oghlan et al. who showed that use-MST solves asymptotically all instances optimally [1, 6].

So far, online versions of the Darp have been investigated for general graphs. The algorithm SmartStart introduced by Ascheuer et al. [7] is 2-competitive w. r. t. completion time, which is optimal. Hauptmeier et al. [3, 8] showed that under appropriate restrictions to the request sequence the Ignore-strategy leads to bounded average and maximum flow time.

Significance and Main Results. Competitive Analysis has been criticized since it often leads to counterintuitive and even worthless results. In fact it is known that there is no constant-competitive algorithm w. r. t. total travel distance for this online problem. The reason is that the offline adversary is too strong to be a sound yardstick for measuring the performance of an online algorithm. A way to overcome this deficiency is to consider randomized online algorithms, which somewhat weakens the adversary. However, in this paper we look at randomized input which means abandoning the focus on unrealistic adversarial instances in favor of a probabilistic model of possible instances. The idea is that this approach leads to statements on typical behavior instead of worst-case behavior.

First we introduce a suitable probabilistic version of the Darp. In doing so, we follow the rationale in [6]: The underlying tree is kept fixed while the requests are generated by a random process. The reason for this is that usually the infrastructure is known and does not change, whereas the requests are unpredictable.

We extend the notion of "competitive ratio" in a natural way to probabilistic input, requiring a good competitive ratio on almost all instances. We analyze an online algorithm based on the offline algorithm use-MST mentioned above and show that it is $(1 + o(1))$-competitive w. r. t. total travel distance in our sense, assuming high load. This shows that the above-mentioned bad behavior of all online algorithms is atypical, at least for request sequences corresponding to our random model. Apart from this it is an example for a situation in which an online algorithm has no big disadvantage compared to a clairvoyant offline algorithm (having no influence on the input instance).

As far as we know this is the first time that a high-probability result for online algorithms is obtained; so far there are only expected-competitiveness results (e. g., [9], [10], [11], [12]).

Although the assumption of high load is unrealistic for real-world applications, our result shows that if there are many requests there is a lot of synergy to exploit. This raises the hope that there are algorithms with a constant probabilistic competitive ratio in realistic load situations.

The remaining part of the paper is structured as follows: Section 2 formally states the variant of the Dial-a-Ride problem we consider, describes the algorithm and defines the random request model. Section 3 contains our main result and its proof, which relies on intermediate results proven in Sections 4 and 5.

2 Problem Statement and Algorithms

The specific properties of our Dial-a-Ride variant allow for a simple graph-theoretic model [2]. Although we will later consider the Dial-a-Ride problem on trees, this model is valid for general graphs. The basic observation is that the time needed to serve a request is minimal if the server travels on a shortest path, since the server can only deal with one request at a time. It remains to find a good ordering of the requests.

Each location is modeled by a vertex of a graph $G = (V, E)$ whose edges are possible interconnections between the locations. The lengths of a connection are provided by an edge length function $d\colon E \to \mathbb{R}_{\geq 0}$. A transportation request is an ordered pair (u, v) (or arc) of locations, its length is the length of a shortest path from u to v. The multiset of all requests is denoted by A. A *tour* is a directed closed walk (sequence of arcs) on the vertex set of our graph, which starts and ends at the depot vertex $o \in V$ and traverses each request arc exactly once. Note that in order to obtain a tour we may have to add arcs connecting the destination location of a request with the source location of the next one. These arcs correspond to *empty rides* of the server. Notice that we do not allow splitting a request arc into successive arcs traversing one edge each, so the non-preemptive-transportations-requirement will be fulfilled. Our goal is to find a tour minimizing the total travel distance.

We say that an edge $\{u, v\}$ (respectively, an arc (u, v)) is *traversed by an arc* (u', v') if the path connecting u' and v' contains $\{u, v\}$ (respectively, the path connecting u and v).

The algorithm USE-MST [2] exploits the following observation: Since a solution to the DARP is a closed walk and we are working on a tree, each edge $\{u, v\}$ has to be traversed from u to v as often as from v to u. Let $m^+(u, v)$ be the number of requests traversing $\{u, v\}$ from u to v and $m^-(u, v)$ be the number of requests going the other way round. Suppose $m^+(u, v) \geq m^-(u, v)$. The multiset B containing exactly $m^+(u, v) - m^-(u, v)$ copies of (v, u) for each edge $\{u, v\}$ is called a *balancing set for* A. It can be seen that the digraph $T' = (V, A \cup B)$ consists of Eulerian components. Due to the construction of B *any* tour has to traverse the arcs in B. One can think of the arcs in B as unavoidable empty rides.

To arrive at a tour it may be necessary to add further arcs to connect the Eulerian components. This can be done by solving a STEINERTREE instance on the *arc-identified graph* arising if all vertices of T' connected by arcs are identified, i. e., all strongly connected components are contracted to a single vertex [2]. Thus any tour consists of request arcs, balancing arcs, and connecting arcs. The algorithm use-MST now works as follows:

1. Compute the balancing set B.
2. Compute a set C of connecting arcs by using the MST-heuristic for the corresponding SteinerTree instance.
3. Return an Eulerian tour of $A \cup B \cup C$.

We now turn to the online problem. To model the online situation we equip a request with a *release time* indicating the time the request becomes known. The task of an online algorithm is to maintain a schedule for the current set of requests, i. e., those requests already known and not yet served, such that the total travel distance of the server is minimized. Each schedule is a solution of the offline instance corresponding to the current set of requests.

The standard way to measure the performance of an online algorithm is the competitive ratio. An online algorithm OlAlg is called *c-competitive* if its cost is at most c times the cost of the optimal offline algorithm on any request sequence ω, i. e., if

$$\textsc{OlAlg}(\omega) \leq c \cdot \textsc{Opt}(\omega).$$

If the ratio c depends on the number of requests m, OlAlg is said to be *non-competitive*.

Suppose we already know a good offline algorithm Alg for Darp. One way to construct an online algorithm from the offline algorithm is to employ the Ignore-strategy [13], which works as follows.

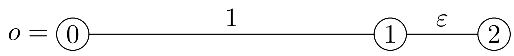
– Initially the server is idle. As soon as requests arrive, we collect the requests and solve the associated Darp instance by employing Alg. The result is a schedule for the requests known so far.
– The server immediately starts executing this schedule. In the meantime, all newly released requests are collected to form the request set for the next reoptimization. This execution of a schedule is called a *phase*.
– If the server finishes its schedule and there are no further requests the server becomes idle. Otherwise, a new schedule is computed as described above and the server executes it.

This online algorithm will be called Ignore(Alg). We will be only concerned with Ignore(use-MST).

The following simple fact shows that deterministic competitive analysis is not of much use for Darp.

Fact 1. *There is no competitive deterministic online algorithm for online* Darp *w. r. t. total travel distance.* □

Proof. We can fool any online algorithm OlAlg on the following transportation network (ε is a small positive constant less than 1):



Now let all requests be of the form $(2, 1)$ and force OLALG to return to the depot for every request, while the offline algorithm serves them all at once. $\square$

The competitive ratio as a measure of performance for online algorithms has been criticized since there are many results like this one. A by now established way to partially circumvent those problems is to consider randomized online algorithms. We will do the opposite: Instead of randomizing the algorithm we look at a randomized input.

Definition 2 (Probabilistic Online model). *A request sequence of length m for a fixed tree $T = (V, E)$ is constructed as follows:*

1. *The m request arcs are drawn independently at random from a symmetric arc distribution $p = (p_{uv})_{u,v \in V}$, i. e., one such that*

$$p_{uv} = p_{vu} \quad \forall u, v \in V.$$

2. *Let $X_1, \ldots, X_m$ be independent identically distributed random variables, which are exponentially distributed with parameter λ. The release time T_i for request i is now given by*

$$T_i = \sum\nolimits_{j=1}^{i} X_j, \qquad 1 \le i \le m.$$

We will use Γ to denote the probability distribution on the request sequences induced by p and λ.

Note that the release times are determined by exponentially distributed interarrival times which is the most basic model for similar considerations in Queuing Theory.

Consider a fixed tree $T = (V, E)$ with edge length function $d\colon E \to \mathbb{R}_{\ge 0}$ and a symmetric request probability distribution $(p_{uv})_{u,v \in V}$. We will denote by μ the expected request length on T and by σ^2 the variance of the request length. We say that we have *high load* if $\lambda > \frac{1}{\mu}$, i. e., requests arrive rapidly.

Last but not least we need a notion of the competitive ratio which takes into account the probabilistic nature of the request sequence. We say that a property holds asymptotically almost surely (a. a. s. for short, cf. [14]) for a random object of size m, if the probability of an object satisfying the property is $1 - o(1)$ as $m \to \infty$.

Definition 3 (Probabilistic competitive ratio). *Let* OLALG *be a (deterministic) online algorithm and suppose $\mathcal{D} = (D_m)_{m \in \mathbb{N}}$ is a family of probability distributions over request sequences, where the parameter m is interpreted as the length of the request sequence, i. e., sequences ω of length m are chosen according to D_m.* OLALG *is said to be c-*competitive w. r. t. *$\mathcal{D}$ if it satisfies*

$$\mathrm{Prob}_{\mathcal{D}}\left[\text{OLALG}(\omega) \le c \cdot \text{OPT}(\omega)\right] = 1 - o(1) \quad \textit{as } m \to \infty.$$

3 A High-level Analysis

In this section we will analyze the algorithm IGNORE(USE-MST) in our setting from a high-level point of view, using results proven in the later sections. For shortness, we will use IGNORE instead of the more precise IGNORE(USE-MST) from now on. Intuitively, we will exploit that the number of requests per phase is high in a high load situation and the requests can be scheduled in a short tour due to synergy effects because the number (and length) of balancing arcs needed will be comparatively small.

To describe the working of IGNORE, we employ the following notation: P is the number of phases, ω_i and m_i are the request subsequence and the number of requests corresponding to phase i, and τ_i is the time needed to execute phase i.

We mentioned earlier that every solution has to use at least the request arcs and the balancing arcs, whereas suboptimal solutions use more connecting arcs than necessary. This gives us a good lower bound for the travel distance of OPT. Denoting by $L(\omega)$ the total length of the requests in ω and by $B(\omega)$ the length of unavoidable balancing arcs for a request sequence ω, we have that

$$\textsc{Opt}(\omega) \geq L(\omega) + B(\omega) \ . \tag{1}$$

Similarly, we can bound IGNORE(ω) as

$$\textsc{Ignore}(\omega) \leq L(\omega) + \sum\nolimits_{i=1}^{P} B(\omega_i) + 2P \sum\nolimits_{e \in E} d(e), \tag{2}$$

where the last term is used as an upper bound for the total length of one MST per phase.

Furthermore, we will exploit the fact that the snapshot problems solved by IGNORE have the same stochastic structure as the offline problem which is easy to see.

We divide the set of phases into *epochs*: For a fixed (small) parameter $\varepsilon \in (0,1)$, phase i belongs to epoch j ($j = 1, \ldots, \frac{1}{\varepsilon} + 1$) if

$$m^{(j-1)\varepsilon} \leq m_i < m^{j\varepsilon}.$$

We will require that starting in epoch 2 the number of requests per phase will be strictly monotonic increasing, which happens a. a. s. This justifies the name "epoch". The first epoch will be treated separately for technical reasons: Here the number of requests in a phase does not depend on m, so we are not able to say anything about its asymptotic behaviour.

In order to arrive at the $(1 + o(1))$-competitive ratio, we will show that the following properties are satisfied a. a. s. if the load is sufficiently high. In the remainder of this paper, we will consider a fixed tree $T = (V, E)$ with edge length function $d\colon E \to \mathbb{R}_{\geq 0}$ and a fixed probability distribution Γ over request sequences as described above. Recall that μ and σ^2 are the expectation and variance of the request length, respectively.

1. The length $L(\omega)$ of the m requests in ω satisfies $L(\omega) \geq (1 - o(1))\mu m$, which is a direct consequence of Chebyshev's inequality.

2. There are $\mathcal{O}\left(\sqrt{m}\log m\right) \subseteq o(m)$ balancing arcs for m requests. Furthermore, there are $\mathcal{O}\left(\sqrt{m_i}\log m_i\right)$ balancing arcs generated by IGNORE in phase i during epochs $2, \ldots, \frac{1}{\varepsilon}+1$ (Proposition 8).
3. There are at most m^{ε} phases with a constant number of requests per phase in epoch 1 and at most $\mathcal{O}(\log m)$ phases in epochs j, $j \geq 2$ (Proposition 10).

Theorem 4. *Let the tree $T = (V, E)$ and the edge length function $d\colon E \to \mathbb{R}_{\geq 0}$ as well as a symmetric request probability distribution $(p_{uv})_{u,v\in V}$ be such that the expected request length μ is larger than 0 and the variance σ^2 is finite. Furthermore, assume that the arrival rate λ satisfies $\lambda > \frac{1}{\mu}$.*

IGNORE(USE-MST) *working on the tree T with request sequences generated by our online model according to $(p_{uv})_{u,v\in V}$ and λ is $\big(1 + o(1)\big)$-competitive w. r. t. Γ.*

Proof. We tacitly assume that the properties 1, 2 and 3 stated above hold. Let Δ denote the diameter of T under d.

We first need a good bound on the length of balancing arcs produced by IGNORE, given by the term $\sum_{i=1}^{P} B(\omega_i)$ in Equation (2). The total number of balancing arcs for the at most m^{ε} phases in epoch 1 is maximized if each phase contains only a single request, which shows that the length of balancing arcs in the first epoch is in $\mathcal{O}(\Delta m^{\varepsilon})$.

Now let us estimate the number of balancing arcs Z generated in later epochs. By definition, every phase i in epoch j serves at most $m^{j\varepsilon}$ requests. As there are at most $\mathcal{O}(\log m)$ phases per epoch, we have

$$\begin{aligned} Z &\in \mathcal{O}(\log m) \sum_{j=2}^{1/\varepsilon+1} \mathcal{O}\left(m^{j\varepsilon/2}\log m_i\right) \\ &\subseteq \mathcal{O}\left(\log^2 m\right) \int_{x=2}^{1/\varepsilon+2} m^{x\varepsilon/2}\,\mathrm{d}x \\ &\subseteq \mathcal{O}\left(m^{1/2+\varepsilon}\log m\right). \end{aligned}$$

Thus the length of the balancing arcs in later epochs is $\mathcal{O}\left(\Delta m^{1/2+\varepsilon}\log m\right)$.

Using the last results in the lower and upper bounds provided by Equations (1) and (2) yields

$$\begin{aligned} \frac{\text{IGNORE}(\omega)}{\text{OPT}(\omega)} &\leq \frac{L(\omega) + \big(\mathcal{O}(\Delta m^{\varepsilon}) + \mathcal{O}\left(\Delta m^{1/2+\varepsilon}\log m\right)\big) + \mathcal{O}(\Delta m^{\varepsilon})}{L(\omega)} \\ &\leq 1 + \frac{\mathcal{O}\left(\Delta m^{\varepsilon} + \Delta m^{1/2+\varepsilon}\log m\right)}{c_\mu m} \end{aligned}$$

where c_μ is some positive constant satisfying $L(\omega) \geq c_\mu m$ a. a. s. We see that this ratio is

$$= 1 + o(1) \quad \text{as } m \to \infty,$$

if we choose an $\varepsilon < \frac{1}{2}$, for instance $\varepsilon = \frac{1}{3}$. $\square$

This result can be used to obtain probabilistic competitiveness results for general graphs, too. The key is to approximate the metric induced by an arbitrary general graph $G = (V, E)$ by a weighted tree, which is possible with distortion $\mathcal{O}(\log|V|)$ [15]. This gives the algorithm GENERAL-USE-MST:

1. Compute a weighted tree T that approximates the metric induced by G with distortion $\mathcal{O}(\log|V|)$.
2. Use USE-MST to solve the problem on T and convert the result to a tour on G.

Corollary 5. *Let $G = (V, E)$ be an arbitrary graph with edge length function $d\colon E \to \mathbb{R}_{\geq 0}$ and $(\tilde{p}_v)_{v \in V}$ be a probability distribution on the vertices. Suppose that the request probability distribution $(p_{uv})_{u,v \in V}$ defined by*

$$p_{uv} := \tilde{p}_u \tilde{p}_v \quad \forall u, v \in V$$

has expected request length $\mu > 0$ and finite variance σ^2. Furthermore, assume that the arrival rate λ satisfies $\lambda > \frac{1}{\mu}$.

IGNORE(GENERAL-USE-MST) *is $\mathcal{O}((1 + o(1)) \log|V|)$-competitive w. r. t. Γ on the graph G.*

Proof. The request distribution is clearly symmetric. Furthermore, the upper bound used in the preceding proof increases by at most a factor of $\mathcal{O}(\log|V|)$. □

A short discussion of the result is in order. First note that the requirement that requests arrive faster than they can be coped with is not a suitable assumption for real-world systems. In fact we do not have reasonable load (see [8]); queuing theorists call such a system *unstable* since the number of requests is ever-increasing if the request sequence continues infinitely.

However, the result is interesting in its own right. It affirms the intuition that if there are many requests there will be synergy effects which can be exploited. Another issue is that the offline algorithm does not have a real advantage over the IGNORE strategy, which is obvious if the constant $\delta := \lambda - \frac{1}{\mu}$ is very large since then most requests arrive in a short interval and IGNORE works essentially like the offline algorithm. But it is also true if δ is very small and there is real online behaviour.

Finally, observe that this result applies also to IGNORE(OPT) since the upper bound for IGNORE(USE-MST) is naturally one for IGNORE(OPT). This implies that asymptotically IGNORE(USE-MST) is as good as IGNORE(OPT).

4 Estimating the Number of Balancing Arcs

We estimate the number Z of balancing arcs needed to balance m randomly generated requests by bounding for each edge e in the tree the number Z_e of balancing arcs traversing e. Clearly,

$$Z = \sum_{e \in E} Z_e.$$

Consider a fixed edge $e = \{u, v\}$ of the tree. What happens to Z_e if a new request is added? The first possibility is that this new request does not use edge e so Z_e does not change. Otherwise, Z_e increases (decreases) by one if the request traverses e from v to u (from u to v). Thus we can express Z_e as

$$Z_e = \left|\sum_{i=1}^{m} X_i(e)\right| \quad \text{for} \tag{3}$$

$$X_i(e) := \begin{cases} 1 & \text{request } i \text{ traverses } \{u,v\} \text{ from } u \text{ to } v \\ -1 & \text{request } i \text{ traverses } \{u,v\} \text{ from } v \text{ to } u \\ 0 & \text{else.} \end{cases}$$

All $X_i(e)$ are identically distributed. To compute the distribution, note that the edge e induces a cut and let $V(u)$ and $V(v)$ be the corresponding vertex sets. If we define

$$\bar{p}_{uv} := \sum_{u' \in V(u), v' \in V(v)} p_{u'v'}$$

for each edge $e = \{u, v\}$ the probability distribution of $X_i(e)$ is given by

$$\text{Prob}\left[X_i(e) = k\right] = \begin{cases} \bar{p}_{uv} & k = 1 \\ \bar{p}_{uv} & k = -1 \\ 1 - 2\bar{p}_{uv} & k = 0. \end{cases}$$

Notice that the symmetry condition on the request distribution implies that $\bar{p}_{uv} = \bar{p}_{vu}$.

Obviously, Z_e behaves similar to the symmetric random walk W_m of length m, i. e., the sum of m i. i. d. random variables taking values 1 and -1 with probability $\frac{1}{2}$ each. We give an equivalent random experiment for generating the vector $\big(X_i(e)\big)_{1 \le i \le m}$ which makes the connection explicit. Intuitively, we first choose the components of the vector which are 0 and fill up the remaining components with a symmetric random walk. Let $p := \bar{p}_{uv}$ and $q := 1 - 2p$.

Definition 6 (Equivalent random model for Z_e). *Construct a random vector $y = \big(Y_i(e)\big)_{1 \le i \le m} \in \{-1, 0, 1\}^m$ as follows:*

- *Choose a set $M_0 \subseteq \{1, \ldots, m\}$ of size $|M_0| =: m_0$ at random, where m_0 is binomially distributed with parameters m and q. The set M_0 determines the zeroes of y: $Y_j(e) := 0$ for all $j \in M_0$.*
- *Choose a realization $w = (W_j)_{j \in [m] \setminus M_0}$ of the symmetric random walk of length $m - m_0$ at random. This determines whether the remaining positions are -1 or 1: $Y_j(e) := W_j$ for all $j \in [m] \setminus M_0$.*

Fact 7. *Fix an integer m. The distributions induced by a random walk according to Equation (3) and by the equivalent random model of Definition 6 on the set $x \in \{-1, 0, 1\}^m$ coincide.* $\square$

We can now directly exploit this fact to estimate $\mathbb{E}[Z]$ via $\mathbb{E}[Z_e]$. Intuitively it is clear that the later should be smaller than the expected travel distance of a symmetric random walk of length m.

Proposition 8. *We have for the number of balancing arcs Z for m requests*

$$Z \leq |E|\sqrt{\frac{2m}{\pi}}\ln m \in \mathcal{O}\left(\sqrt{m}\log m\right) \quad \textit{a. a. s.}$$

Starting from epoch 2, every request subsequence ω_i corresponding to phase i a. a. s. fulfills $B(\omega_i) \in \mathcal{O}\left(\sqrt{m_i}\log m_i\right)$.

Proof. We compute $\mathbb{E}[Z_e]$ by using the equivalent random model for Z_e. Let W_m denote the symmetric random walk of length m.

By using the Binomial Theorem and the fact that $\mathbb{E}[|W_m|] \leq \mathbb{E}[|W_{m+k}|]$ for $k \geq 0$ we have the estimate

$$\begin{aligned}
\mathbb{E}[Z_e] = \mathbb{E}\left[\left|\sum\nolimits_{i=1}^{m} Y_i\right|\right] &= \sum\nolimits_{m_0=0}^{m}\binom{m}{m_0}q^{m_0}(2p)^{m-m_0}\mathbb{E}[|W_{m-m_0}|] \\
&\leq \mathbb{E}[|W_m|]\sum\nolimits_{m_0=0}^{m}\binom{m}{m_0}q^{m_0}(2p)^{m-m_0} \\
&= \mathbb{E}[|W_m|] \sim \sqrt{\frac{2m}{\pi}}.
\end{aligned}$$

In the last step we exploited the asymptotics of the expected distance of the symmetric random walk [16], namely $\mathbb{E}[|W_m|] \sim \sqrt{2m/\pi}$, which means that $\lim_{m\to\infty}\frac{\mathbb{E}[|W_m|]}{\sqrt{2m/\pi}} = 1$.

To obtain the sharp concentration result, we want to apply Azuma's inequality (see [14]). We have to show that the function $f(r_1,\ldots,r_m) := \sum_{e\in E}|Z_e|$ satisfies the required Lipschitz-condition. Suppose that request r_i is replaced by request r_i'. How does this influence Z_e? We have already seen that Z_e changes by at most 1 if a request is added and the same holds if a request is deleted. Thus, Z_e increases by at most 2, so we get

$$|f(r_1,\ldots,r_{i-1},r_i,r_{i+1},\ldots,r_m) - f(r_1,\ldots,r_{i-1},r_i',r_{i+1},\ldots,r_m)| \leq 2|E|.$$

Applying Azuma's inequality now yields

$$\begin{aligned}
\mathrm{Prob}[Z \geq \mathbb{E}[Z] + t] &\leq \exp\left(-\frac{t^2}{2\sum_1^m 4|E|^2}\right) \\
&= \exp\left(-\frac{t^2}{8|E|^2 m}\right).
\end{aligned}$$

Choosing $t := \sqrt{m}\ln m$ shows that $Z \leq \mathbb{E}[Z]\ln m \in \mathcal{O}\left(\sqrt{m}\log m\right)$ a. a. s., namely with probability at least $1 - \frac{1}{m^2}$.

Consider the phases in epochs j, $j \geq 2$. In all of these phases there are at least m^{ε} requests and Proposition 10 states that there are at most $\mathcal{O}(\log m)$ of them. Thus the probability that none of them violates the condition $B(\omega_i) \in \mathcal{O}\left(\sqrt{m_i}\log m_i\right)$ is at least

$$\left(1 - \frac{1}{m^{2\varepsilon}}\right)^{\mathcal{O}(\log m)} \geq 1 - \mathcal{O}\left(\log m\right) \cdot \frac{1}{m^{2\varepsilon}} = 1 - o(1) \quad \text{as } m \to \infty$$

by Bernoulli's inequality. □

5 There are Many Requests per Phase

An important issue in this analysis is the fact that there are not too many phases, because this keeps the number of balancing arcs generated by IGNORE small. To prove this fact we will need that the release rate is *larger* than the rate at which requests can be finished. In that case the number of requests should grow (on average) from phase to phase. There are two points to consider: First, we have to show that once there are "lots of requests" the probability is high that this property is also satisfied in the next phase. Second, we have to ensure that it will not take too many phases until there are the first time "lots of requests".

The following lemma bounds the probability that in an interval of length τ_i significantly less requests arrive than are expected. We will use the well-known result from Queuing Theory that the number of requests generated by a process with exponentially distributed interarrival times is Poisson-distributed [17].

Lemma 9. *Recall that m_i denotes the number of requests in phase i and that τ_i is the time needed to serve these requests. Fix an $\alpha \in (0, 1)$.*

$$\text{Prob}\left[m_{i+1} \leq \alpha\lambda\tau_i \mid m_i = k\right] \in \mathcal{O}\left(\mathrm{e}^{-\lambda\tau_i}\right).$$

Note that τ_i implicitly depends on k.

Proof. Some calculations which can be found in the appendix. □

Proposition 10. *Let the arrival rate be $\lambda = \frac{1}{\mu} + \delta$ for some positive constant δ and fix some $\varepsilon \in (0, 1)$.*

Then the first epoch a. a. s. consists of at most m^ε phases with at least a constant number of requests per phase. Furthermore, in each epoch j, $j > 2$, the number of requests increases from phase to phase (with the possible exception of the last phase) and thus the epoch consists of $\mathcal{O}(\log m)$ phases a. a. s.

Proof. We first show that after time m^ε we have a. a. s. $\Omega(m^\varepsilon)$ requests. The fact that $m_i \geq m^\varepsilon$ together with Lemma 9 can be used to establish $m_{i+1} > m_i$ a. a. s. This allows us to conclude that there are at most $\mathcal{O}(\log m)$ phases, which in turn can be used to show that $m_{i+1} > m_i$ for all i in the later epochs. Details can be found in the appendix. □

6 Conclusions

We have introduced a novel notion of the competitive ratio for probabilistic input sequences and we showed a first non-trivial result for this new notion. Of course, the high-load assumption is somewhat artificial. The behavior in a normal load setting is more complicated and it will be a challenging task to do a similar analysis. We believe that a careful analysis will provide handholds for analyzing different objective functions.

Acknowledgements I want to thank my colleague Sven O. Krumke for some suggestions and fruitful discussions.

References

1. Coja-Oghlan, A., Krumke, S.O., Nierhoff, T.: Scheduling a server on a caterpillar network - a probabilistic analysis. In: Proceedings of the 6th Workshop on Models and Algorithms for Planning and Scheduling Problems. (2003)
2. Frederickson, G.N., Guan, D.J.: Nonpreemptive ensemble motion planning on a tree. Journal of Algorithms **15** (1993) 29–60
3. Hauptmeier, D.: Online algorithms for transport systems. Technische Universität Berlin (1999) Diplom Thesis.
4. Atallah, M.J., Kosaraju, S.R.: Efficient solutions to some transportation problems with application to minimizing robot arm travel. SIAM Journal of Computation **17** (1988) 819–869
5. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. SIAM Journal of Computation **7** (1978) 178–193
6. Coja-Oghlan, A., Krumke, S.O., Nierhoff, T.: A heuristic for the stacker crane problem on trees which is almost surely exact. In: Proceedings of the 14th Annual International Symposium on Algorithms and Computation. Volume 2906 of Lecture Notes in Computer Science., Springer (2003) 605–614
7. Ascheuer, N., Krumke, S.O., Rambau, J.: Online Dial-a-Ride problems: Minimizing the completion time. In: Proceedings of the 17st Symposium on Theoretical Aspects of Computer Science. Volume 1770 of Lecture Notes in Computer Science., Springer (2000) 639–650
8. Hauptmeier, D., Krumke, S.O., Rambau, J.: The online Dial-a-Ride problem under reasonable load. In: CIAC 2000. Volume 1767 of Lecture Notes in Computer Science., Springer (2000) 125–136
9. Becchetti, L., Leonardi, S., Marchetti-Spaccamela, A., Schäfer, G., Vredeveld, T.: Average case and smoothed competitive analysis for the multi-level feedback algorithm. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science. (2003) 462–471
10. Fujiwara, H., Iwama, K.: Average-case competitive analyses for ski-rental problems. In: ISAAC 2002. Volume 2518 of Lecture Notes in Computer Science., Springer (2002) 476–488
11. Scharbrodt, M., Schickinger, T., Steger, A.: A new average case analysis for completion time scheduling. In: Proceedings of the 34th ACM Symposium on Theory of Computing, Montréal, QB. (2002) 170–178

12. Souza-Offtermatt, A., Steger, A.: The expected competitive ratio for weighted completion time scheduling. In: Proceedings of the 21st Symposium on Theoretical Aspects of Computer Science. Volume 2996 of Lecture Notes in Computer Science. (2004) 620–631
13. Ascheuer, N., Krumke, S.O., Rambau, J.: The online-transportation problem: Competitive scheduling of elevators. Technical report, Zuse Institute Berlin (1998) Preprint SC 98-34.
14. Janson, S., Łuczak, T., Ruciński, A.: Random Graphs. Interscience Series in Discrete Mathematics and Optimization. Wiley (2000)
15. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. In: Proceedings of the 35th ACM Symposium on Theory of Computing. (2003) 448–455
16. Weisstein, E.W.: CRC Concise Encyclopedia of Mathematics. CRC Press (1998)
17. Kleinrock, L.: Queuing Systems. Volume 1: Theory. John Wiley & Sons (1975)

A Proofs

Lemma 9. *Recall that m_i denotes the number of requests in phase i and that τ_i is the time needed to serve these requests. Fix an $\alpha \in (0,1)$.*

$$\mathrm{Prob}\left[m_{i+1} \leq \alpha\lambda\tau_i \mid m_i = k\right] \in \mathcal{O}\left(\mathrm{e}^{-\lambda\tau_i}\right).$$

Note that τ_i implicitly depends on k.

Proof. Using the distribution function of the Poisson distribution we can express the left hand side as

$$\begin{aligned}\mathrm{Prob}\left[m_{i+1} \leq \alpha\lambda\tau_i \mid m_i = k\right] &\leq \sum\nolimits_{i=0}^{\alpha\lambda\tau_i} \frac{(\lambda\tau_i)^i}{i!}\mathrm{e}^{-\lambda\tau_i} \\ &= \mathrm{e}^{-\lambda\tau_i} \sum\nolimits_{i=0}^{\alpha\lambda\tau_i} \frac{(\lambda\tau_i)^i}{i!}.\end{aligned}$$

To prove the lemma we need to show that the sum at the right hand side is in $\mathcal{O}\left(\mathrm{e}^{c\lambda\tau_i}\right)$ for some $c < 1$. To simplify our computations let $x := \lambda\tau_i$ and $i_0 := \alpha x$. Since $\alpha < 1$ we know that the summands are increasing which leads to the bound

$$\begin{aligned}\sum\nolimits_{i=0}^{i_0} \frac{x^i}{i!} &\leq i_0\left(\frac{x^{i_0}}{i_0!}\right) + 1 \\ &\sim i_0 \frac{x^{i_0}\mathrm{e}^{i_0}}{\sqrt{2\pi i_0}\,(\alpha x)^{i_0}} \\ &= \sqrt{\frac{i_0}{2\pi}}\left(\frac{\mathrm{e}}{\alpha}\right)^{i_0} \\ &= \sqrt{\frac{i_0}{2\pi}}\mathrm{e}^{i_0(1-\ln\alpha)} \\ &\leq \mathrm{e}^{\alpha(1-\ln\alpha)x+\mathcal{O}(1)} \in \mathcal{O}\left(\mathrm{e}^{cx}\right).\end{aligned}$$

The function $f(\alpha) := \alpha(1-\ln\alpha)$ is strictly increasing in $(0,1)$ and $\lim_{\alpha\nearrow 1} f(\alpha) = 1$, so the constant c is smaller than 1. $\square$

Proposition 10. *Let the arrival rate be* $\lambda = \frac{1}{\mu} + \delta$ *for some positive constant* δ *and fix some* $\varepsilon \in (0, 1)$*.*

Then the first epoch a. a. s. consists of at most m^ε *phases with at least a constant number of requests per phase. Furthermore, in each epoch* j*,* $j > 2$*, the number of requests increases from phase to phase (with the possible exception of the last phase) and thus the epoch consists of* $\mathcal{O}(\log m)$ *phases a. a. s.*

Proof. Let us first examine how many requests are unserved at time $t(m)$. Denote by $N_{t(m)}$ the number of requests already released at time $t(m)$ and by $D_{t(m)}$ the number of requests already served. Clearly, $N_{t(m)}$ is Poisson-distributed with parameter $\lambda t(m)$, so we have $\mathbb{E}\left[N_{t(m)}\right] = \lambda t(m)$. Invoking Chebyshev's inequality with $t := \sqrt{t(m)} \log m$ tells us that $N_{t(m)} \geq (1 - o(1))\lambda t(m)$ a. a. s.

We already noted that m requests need a. a. s. at least time $(1 - o(1))m\mu$ to be served. If $t(m)$ is sufficiently large we can assume that each request takes time $(1 - o(1))\mu$, providing us with the bound $D_{t(m)} \leq \frac{t(m)}{(1-o(1))\mu} = \frac{(1+o(1))}{\mu} t(m)$. Combining both estimates we see that at time $t(m)$ there are at least

$$\begin{aligned} N_{t(m)} - D_{t(m)} &\geq (1 - o(1))\lambda t(m) - \frac{(1 + o(1))}{\mu} t(m) \\ &= t(m) \left((1 - o(1)) \left(\frac{1}{\mu} + \delta \right) - \frac{(1 + o(1))}{\mu} \right) \\ &= t(m) \left((1 - o(1))\delta - \frac{o(1)}{\mu} \right) \\ &= \delta(1 - o(1))t(m) \end{aligned}$$

requests in the system. Choosing $t(m) := m^\varepsilon$ we have that after at most m^ε phases with at least constantly many requests there is a first phase with $\Omega(m^\varepsilon)$ requests.

It remains to establish that once IGNORE-phases feature m^ε requests (i. e., as of epoch 2) the sequence of requests in the phases is strictly monotonic increasing a. a. s., that is $m_{i+1} > m_i$ for all phases $i + 1 \neq P$ in epoch j, $j \geq 2$. Assume for the moment that there are at most $\mathcal{O}(\log m)$ phases in those epochs and that the request generation process is not stopped after the mth request.

Suppose phase i_0 is the first phase of epoch 2. Since the length of k requests is a. a. s. $\geq (1 - o(1))\mu k$, we see that $\tau_{i_0} \geq (1 - \varepsilon_0)\mu m_{i_0}$ a. a. s. for some suitable $\varepsilon_0 \in (0, 1)$. We want to invoke Lemma 9 to guarantee that $m_{i_0+1} > m_{i_0}$ a. a. s., so we need an α such that $\alpha\lambda\tau_{i_0} > m_{i_0}$, or equivalently

$$\alpha \left(\frac{1}{\mu} + \delta \right) (1 - \varepsilon_0)\mu m_{i_0} > m_{i_0}.$$

From the last inequality we get the condition

$$\alpha > \frac{1}{(1 + \delta\mu)(1 - \varepsilon_0)}$$

for an appropriate choice of α. On the other hand, α has to be less than 1, leading to another condition on ε_0

$$1 > \frac{1}{(1+\delta\mu)(1-\varepsilon_0)} \qquad \Longleftrightarrow \qquad 1-\varepsilon_0 > \frac{1}{1+\delta\mu}.$$

There are values for α and ε_0 which obey all conditions so we get that $m_{i_0+1} > m_{i_0}$ a. a. s.

In the preceding argument we did not explicitly use that we dealt with the *first* phase of epochs $2, \ldots, \frac{1}{\varepsilon}+1$, so this choice of α and ε_0 indeed guarantees $m_{i+1} > m_i$ for all $i+1 > i_0$. We still have to consider the probability that we have this kind of "success" for *all* phases. By Bernoulli's inequality, the probability that $m_{i+1} > m_i$ for all $i \geq i_0$ is at least

$$\left(1-\mathcal{O}\left(\mathrm{e}^{-\lambda m^{\varepsilon}}\right)\right)^{\mathcal{O}(\log m)} \geq 1-\mathcal{O}(\log m)\cdot\mathcal{O}\left(\mathrm{e}^{-\lambda m^{\varepsilon}}\right) = 1-o(1) \quad \text{as } m\to\infty.$$

Similarly, the probability that $\tau_i \geq (1-\varepsilon_0)\mu m_i$ for all $i \geq i_0$ is at least

$$\left(1-\sigma^2 m^{-\varepsilon/3}\right)^{\mathcal{O}(\log m)} \geq 1-\mathcal{O}(\log m)\cdot\sigma^2 m^{-\varepsilon/3} = 1-o(1) \quad \text{as } m\to\infty.$$

Here we used the fact that the length L_m of m requests is at least $\mu m - m^{2/3}$ with probability at most $\sigma^2 m^{-1/3}$ (Chebyshev inequality).

So far we have seen that the number of requests per phase is a. a. s. strictly monotonic increasing provided that there are at most $\mathcal{O}(\log m)$ phases in the last epochs and there are infinitely many requests. The number of requests per phase grows by a factor of $\alpha\lambda(1-\varepsilon_0)\mu > 1$ so there are at most $\mathcal{O}(\log m)$ phases in an epoch. This holds also if we consider only the first m requests of the infinitely many requests, since then only the last phase P may violate the increasing-condition. Since there are not more than $\frac{1}{\varepsilon}+1$ epochs the number of phases is indeed $\mathcal{O}(\log m)$ in total. $\square$