Department of Applied Mathematics
Faculty of EEMCS

University of Twente
The Netherlands

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl
www.math.utwente.nl/publications

Memorandum No. 1675

Online matching on a line

B. Fuchs,[1]
W. Hochstättler[2] and W. Kern

May, 2003

ISSN 0169-2690

[1]Zentrum für Angewandte Informatik Köln, Universität zu Köln, Weyertal 80, D-50931 Köln
[2]Department of Mathematics, BTU Cottbus, Postfach 10 13 44, D-03013 Cottbus

Online Matching On a Line

Bernhard Fuchs*
Zentrum für Angewandte Informatik Köln
Universität zu Köln, Weyertal 80, D-50931 Köln

Winfried Hochstättler
Department of Mathematics, BTU Cottbus
Postfach 10 13 44, D-03013 Cottbus

Walter Kern
Department of Applied Mathematics
University of Twente, P.O.Box 217, NL-7500 AE Enschede

April 1, 2003

Abstract

We prove a lower bound $\rho \geq 9.001$ for the competitive ratio of the so-called online matching problem on a line. As a consequence, the online matching problem is revealed to be strictly more difficult than the "cow problem".

Keywords: online algorithms, competitive analysis, matching
MSC codes: 68M20, 68Q25

1 Introduction

We consider a special class of online server problems, where a number of servers, located on the real line, is to serve a sequence of requests $r_1, r_2, \ldots, r_k \in$

*supported by Deutsche Forschungsgemeinschaft, Graduiertenkolleg Scientific Computing, GRK 192/5-02

$\mathbb{R}$. In contrast to classical server problems (cf, e.g. [3]), however, each server can serve at most one request. So the optimal offline solution is the min cost matching of the requests into the set of server positions s_i. The problem is therefore also known as the *online matching problem on a line* ([5]). As an application, consider a ski rental with different ski lengths $s_1, s_2, \ldots$ at its disposal to meet requested lengths $r_1, r_2, \ldots$ of entering clients.

For notational convenience, we consider a "universal" instance with infinitely many servers, one at each integer $s \in \mathbb{Z}$. One may equally well consider finite versions with servers at positions $s_1, \ldots, s_n \in \mathbb{R}$, requests $r_1, \ldots r_k \in \mathbb{R}$ and $k \leq n$ (or even $k = n$). These are, however, easily seen to be of approximately the same difficulty: Any ρ-competitive algorithm for one version with $\rho < \bar{\rho}$ implies a $\tilde{\rho}$-competitive version for the other with $\tilde{\rho} < \bar{\rho}$.

An online matching algorithm is *ρ-competitive* if, after serving $r_1, \ldots, r_t$ ($t \in \mathbb{N}$), the current length L of the online matching constructed so far is at most ρ times the current optimal matching cost. It is a challenging open question to prove or disprove the existence of ρ-competitive online algorithms with finite competitive ratio ρ.

The basic difficulty for an online algorithm is to decide which server to use for matching a new request r. There are essentially two choices: Either the server s_- that is closest to r from left or the server s_+ that is closest to r from right (among those servers that are currently still unmatched). Indeed, serving r from a server at $s < s_-$ can be interpreted as moving s to s_- and serving r from s_-.

Assume, e.g., the first $2m_0$ requests are at $r = 0, \pm 1, \pm 2, \ldots, \pm(m_0 - 1), 0$. The first $2m_0 - 1$ requests will then be served from the servers at these positions, whereas the last request $r = 0$ will be served, say, from $s = -m_0$. Assume the following requests $r_{2m_0+1}, r_{2m_0+2}, \ldots$ are then exactly at the positions where a server has just been moved off to serve the previous request. So $r_{2m_0+1} = -m_0$ etc. In order to stay ρ-competitive, the online algorithm may first serve a number of requests from left, but must eventually *switch* to serving some request $r = i \leq -m_0$ from right, i.e., from $s = m_0$. (Indeed, $|i| \leq \rho/2m_0$). It may then continue to serve a number of requests from right, but eventually it will have to switch again, serving some request $r = j \geq m_0$ from left etc. Thus the online algorithm basically must behave like the famous cow searching for a bridge to cross the river ([1], [4]). We therefore refer to the request sequence constructed as above as a *cow sequence* with parameter m_0, started at $r = 0$.

This analogy yields a lower bound of $\rho \geq 9$ for the competitive ratio of any online algorithm for matching on a line (even for cow sequences), cf. [1] or section 2. The main purpose of our paper is to slightly improve this bound to $\rho \geq 9.001$. Since a 9-competitive algorithm for the "cow problem" is known, our result proves the online matching problem to be strictly more difficult than the cow problem. In Section 4 we analyze online algorithms based on so-called *work functions* and show that they have infinite competitive ratio.

2 Cow Sequences

Consider an online algorithm for the matching problem on a line and assume it has already served requests $r_1, \ldots, r_k \in \mathbb{Z}$. We denote by L the (length of) the matching constructed so far and refer to it as the *current travel length*. M^* denotes the (length of) the current optimal matching from $R = \{r_1, \ldots, r_t\}$ into $\mathbb{Z}$. In addition, we introduce the *current matching* M: Assume that the online algorithm has served the currently known set of requests $R = \{r_1, \ldots, r_t\}$ from servers $S = \{s_1, \ldots, s_t\}$. Then M is the (length of) the optimal matching from S to R. We stress that, in general, this is different from both L and M^*.

As an example, consider a cow sequence as in Section 1 and assume that the online algorithm switches at $r = -i$ to serving from right and then continues serving $r = m_0, r = m_0 + 1, \ldots, r = j - 1$ from right. The current matching M is then the assignment $m_0 \mapsto 0, m_0+1 \mapsto m_0, \ldots, j \mapsto j-1$ (cf. Figure 1).

Figure 1: The current matching M ($m_0 = 1$)

In the situation indicated in Figure 1 we have $M = j, L = 2i + j$ and $M^* = i + 1$ (assuming that $j > i$). We always indicate unused servers by $\circ$.

We use current matchings to analyze the behaviour of a ρ-competitive algorithm for the matching problem (and provide a new proof for the lower bound $\rho \geq 9$ on cow sequences). When the online algorithm serves a cow sequence, we let $M_k, k \geq 1$, denote the current matching immediately after the k-th switch (cf. Figure 2).

After the k-th switch, when the current matching is M_k, the online algorithm

Figure 2: The current matching M_k

has travelled $L_1 = 2M_2 + M_1 - 2$ if $k = 1$ and

$$L_k = 2\left(\sum_{i=2}^{k+1} M_i - 1\right) + M_k = 2\sum_{i=2}^{k-1} M_i + 3M_k + 2M_{k+1} - 2k, \text{ for } k \geq 2. \quad (1)$$

The standard online algorithm for serving cow sequences is based on the *doubling technique*, switching between left and right so that $M_k = 2M_{k-1}$ holds for $k \geq 2$. This in particular guarantees that, after each switch, the current matching $M = M_k$ is the current optimal assignment $M^* = M_k^*$ (and M stays optimal until it exceeds M_{k+1}). Furthermore, by induction we have

$$L_k = 9M_k - 4M_1 - 2k. \quad (2)$$

Thus, the doubling technique is 9-competitive for serving cow sequences.

To see that the factor 9 is best possible, consider an arbitrary online algorithm for serving cow sequences, producing current matchings M_k and travel lengths L_k after the k-th switch. Let σ_k and α_k be such that

$$L_k = (9 - \sigma_k)M_k \qquad \text{and} \qquad M_{k+1} = (1 + \alpha_k)M_k.$$

Remark 1 *The doubling technique would correspond to $\alpha_k = 1$, $k \geq 1$. Here, we have $\alpha > -1$ (by definition). In particular, α_k may be negative, so that M_k is no longer guaranteed to be the current optimal assignment for all $k \geq 1$.*

We introduce the *potential*

$$\Phi_k := \sigma_k + 2\alpha_k, \qquad k \geq 1.$$

Remark 2 *For a 9-competitive algorithm, $\sigma \geq 0$ indicates the current "length credit" (relative to the current M) and α can be interpreted as the credit we have gained by searching a region of size $(1 + \alpha)M$ on the opposite side.*

We calculate

$$\Phi_1 = 9 - \frac{M_1 + 2M_2 - 2}{M_1} + 2\alpha_1 = 6 + \frac{2}{M_1} = 6 + \frac{2}{m_0} \approx 6$$

and

$$\Phi_2 = 9 - \frac{3M_2 + 2M_3 - 4}{M_2} + 2\alpha_2 = 4 + \frac{4}{M_2} \approx 4,$$

assuming m_0 is chosen sufficiently large.

Furthermore, observe that a ρ-competitive algorithm must necessarily produce exponentially growing M_k's, in the sense that, for example, certainly $M_{k+2\lceil\rho\rceil} \geq 2M_k$ must hold. Thus, $\frac{k}{M_k}$ can be made arbitrarily small by an appropriately large choice of m_0. On the other hand, (1) implies that a ρ-competitive algorithm must certainly maintain $M_k \leq \frac{\rho}{2} M_{k-1}$. This gives a rough upper bound on Φ_k. For $k \geq 3$ we derive

$$\begin{aligned}(9 - \sigma_k)M_k = L_k &\geq 2M_{k-1} + 3M_k + 2(1 + \alpha_k)M_k - 2k \\ &\geq \left(\frac{4}{\rho} + 5\right) M_k + 2\alpha_k M_k - 2k.\end{aligned}$$

Dividing by M_k we arrive at

$$\Phi_k \leq 4 - \frac{4}{\rho} + \frac{2k}{M_k} < 4 - \frac{2}{\rho}, \tag{3}$$

for m_0 sufficiently large.

To establish a recursion for Φ_k, α_k and σ_k we compute from (1) that

$$(9 - \sigma_{k+1})M_{k+1} - (9 - \sigma_k)M_k = L_{k+1} - L_k = 2M_{k+2} + M_{k+1} - M_k - 2.$$

Substituting $M_{k+1} = (1 + \alpha_k)M_k$ and $M_{k+2} = (1 + \alpha_{k+1})(1 + \alpha_k)M_k$ gives

$$\begin{aligned}(\sigma_{k+1} + 2\alpha_{k+1})(1 + \alpha_k) &= 6\alpha_k + \sigma_k - 2 + \frac{2}{M_k} \\ &= (\sigma_k + 2\alpha_k)(1 + \alpha_k) - (\alpha_k\sigma_k + 2(1 - \alpha_k)^2) + \frac{2}{M_k}.\end{aligned}$$

Dividing by $1+\alpha_k$, we arrive at

$$\Phi_{k+1} = \Phi_k - \Delta_k + \frac{2}{M_k} \quad \text{with} \quad \Delta_k = \frac{\alpha_k\sigma_k + 2(1-\alpha_k)^2}{1+\alpha_k}. \tag{4}$$

Remark 3 *The exponential growth rate of the M_k's ensures that $\sum \frac{2}{M_k}$ can be made arbitrarily small, so that the update $\Phi_{k+1} = \Phi_k - \Delta_k$ would give approximately correct Φ values.*

It is now easy to see that $(9-\varepsilon)$-competitive algorithms for serving cow sequences (and hence, *a fortiori*, for matching on a line) cannot exist: Such an algorithm would maintain $\sigma_k \geq \varepsilon$. Note, that by (1) we must have $\varepsilon \leq 6$. Therefore, by (4), Δ_k as a function of α_k attains its minimum in $]-1,\infty[$ for non-negative α and this implies

$$\Delta_k \geq \frac{1}{3}\varepsilon + \frac{\frac{1}{3}\alpha_k\varepsilon + \frac{1}{3}\varepsilon(\alpha_k-1) + 2(\alpha_k-1)^2}{1+\alpha_k} \geq \frac{1}{3}\varepsilon \tag{5}$$

in each step. The last inequality follows since the minimum of the denominator of the fraction is attained at $\alpha_k = 1 - \frac{1}{6}\varepsilon$.

So the update $\Phi_{k+1} = \Phi_k - \Delta_k$, and, similarly, $\Phi_{k+1} = \Phi_k - \Delta_k + \frac{2}{M_k+1}$, would yield $\lim_{k\to\infty}\Phi_k \to -\infty$, whereas $\Phi_k = \sigma_k + \alpha_k \geq \varepsilon + 2(-1)$ must hold, a contradiction.

Our approach also reveals that any 9-competitive algorithm must asymptotically follow the doubling technique when serving a cow sequence. Indeed our argument above yields for $\sigma_k \geq 0$ that $\Delta_k \geq 0$ in (4) and even more $\sum_{j\geq k}\Delta_j$ must converge to zero as k tends to ∞. This can only happen when $\alpha_k \to 1$ and $\sigma_k \to 0$.

The main difficulty in analyzing $(9+\varepsilon)$ competitive algorithms serving a cow sequence is due to the fact that $\sigma < 0$ and hence $\Delta < 0$ may occur, causing an *increase* of the potential. The following lemma bounds Δ from below and gives sufficient conditions for Δ being significantly positive.

Lemma 1 *For a $(9+\varepsilon)$-competitive algorithm serving a cow sequence with m_0 sufficiently large and $0 \leq \varepsilon \leq \frac{1}{4}$ we have in iteration $k \geq 3$*

1. $\Delta \geq -\varepsilon$
2. $\alpha \leq 1 - \frac{3}{4}\sqrt{\varepsilon} \Rightarrow \Delta \geq \frac{1}{16}\varepsilon$

3. $\sigma \geq \varepsilon \Rightarrow \Delta \geq \frac{1}{3}\varepsilon$

4. $\Phi \leq 2 - 2\sqrt{\varepsilon} \Rightarrow \Delta \geq \frac{1}{16}\varepsilon$.

Proof: By (3) we have for $k \geq 3 : \Phi < 4 - \frac{2}{9+\varepsilon} \leq 4 - \frac{1}{5}$. Thus, in case $-1 < \alpha < 0$ we get

$$\Delta(\alpha) = \frac{\alpha(\Phi - 4) + 2}{1 + \alpha} > \frac{2}{1 + \alpha} > 2.$$

Hence, in the following, we may assume $\alpha \geq 0$.

By (4), $\Delta \geq \frac{\alpha}{\alpha+1}\sigma \geq \frac{\alpha}{\alpha+1}(-\varepsilon) \geq -\varepsilon$. This proves 1.

If $0 \leq \alpha \leq 1 - \frac{3}{4}\sqrt{\varepsilon}$,

$$\Delta(\alpha) = \frac{\alpha\sigma + 2(1-\alpha)^2}{1+\alpha} \geq \frac{-\varepsilon + 2 \cdot \frac{9}{16}\varepsilon}{1+\alpha} \geq \frac{1}{16}\varepsilon,$$

which proves 2.

3. has been proven in (5). Finally, $0 \leq \varepsilon \leq \frac{1}{4}$ yields $\varepsilon \leq \frac{\sqrt{\varepsilon}}{2}$. Thus, $\Phi \leq 2 - 2\sqrt{\varepsilon}$ implies $\alpha \leq 1 - \frac{3}{4}\sqrt{\varepsilon}$. □

3 More Cows

The basic idea for proving a lower bound $\rho \geq 9 + \varepsilon$ for online matching is to run two (or more) cow sequences. Assume, we have two "cows" with current matchings M and $\bar{M}$, directed away from each other, as indicated in Figure 3.

Figure 3: Two cows in opposition

Assume that we continue the first cow sequence, i.e. we request $r = M, M+1$ etc. Furthermore, assume the online algorithm serves all these request from right, thus extending M to a point "beyond the second cow" (cf. Figure 4 a)) until it switches back to M' (cf. Figure 4 b)).

This results in a *combined cow* (cf. Figure 4 b)) in the sense that, when the request sequence is continued with $r = -M', -M' - 1, \ldots$, the online

Figure 4: Combining two cows

algorithm behaves like if the current matching was $\tilde{M} = M' + \bar{M}$ and can be analyzed like a "simple cow".

In absence of the second cow, the new potential of the first cow (after switching back to M') would be Φ'. The effect of "eating up the second cow" is that, under certain circumstances, the potential $\tilde{\Phi}$ of the combined cow is smaller than Φ'.

The parameters $\tilde{\alpha}$ and $\tilde{\sigma}$ of the combined cow can be computed from

$$(9 - \tilde{\sigma})\tilde{M} = \tilde{L} = L' + \bar{L} = (9 - \sigma')M' + (9 - \bar{\sigma})\bar{M}$$

and the "total range equality"

$$(2 + \alpha')M' = (2 + \tilde{\alpha})\tilde{M}.$$

Thus we get

$$\tilde{\sigma}\tilde{M} = \sigma' M' + \bar{\sigma}\bar{M} \text{ and } \tilde{\alpha}\tilde{M} = \alpha' M' - 2\bar{M}.$$

The combined potential $\tilde{\Phi}$ is thus

$$\tilde{\Phi} = \frac{M'}{\tilde{M}}\Phi + \frac{\bar{M}}{\tilde{M}}(\bar{\sigma} - 4), \tag{6}$$

which is strictly less than Φ', for example, when $\bar{\sigma} < 4$.

Note that, since $\Phi' \leq \Phi + \varepsilon + \frac{2}{M'}$ (cf. Lemma 1 1.), we may expect that even $\tilde{\Phi} < \Phi$ holds. This is the basic idea of our approach: We run a cow sequence as long as the potential decreases significantly, say $\Delta \geq \frac{\varepsilon}{16}$. When this is no longer guaranteed, i.e., $\Delta < \frac{\varepsilon}{16}$ occurs, we start a little "second cow" to be eaten up in the next step, so that the potential decreases nonetheless. Eventually, the potential will thus drop below $2-2\sqrt{\varepsilon}$, which guarantees $\Delta \geq \frac{\varepsilon}{16}$ by Lemma 1. From this point on, the potential will decrease automatically, i.e., Φ would decrease to $-\infty$, a contradiction.

To work this out in detail, consider a $(9+\varepsilon)$-competitive algorithm for matching on a line with, say, $\varepsilon = 0.001$. We start a cow sequence at $r = 0$ and

sufficiently large m_0. As long as $\Delta \geq \frac{\varepsilon}{16}$, we continue the sequence. Eventually, since $\Phi > -\varepsilon - 2$, $\Delta < \frac{\varepsilon}{16}$ must occur, implying

$$\sigma \leq \frac{\varepsilon}{5} \qquad \text{and} \qquad \alpha \geq 1 - \sqrt{\varepsilon}$$

by Lemma 1.

Assume w.l.o.g. that the current matching $M = M_k$ points to the left as in Figure 3. We then start a second cow at $r = \lceil 1.1M \rceil$ with $\bar{m}_0 = \lceil \varepsilon M \rceil$. The total length credit that we inherit from the first cow is $(\sigma + \varepsilon)M \leq \frac{6}{5}\varepsilon M$. So the second cow is certainly bound to be 11-competitive. Assume it produces current matchings $\bar{M}_k$. Then

$$\bar{M}_1 = \lceil \varepsilon M \rceil \qquad \text{and} \qquad \bar{M}_2 \leq 5\lceil \varepsilon M \rceil,$$

since $\bar{L}_1 = 2\bar{M}_2 + \bar{M}_1 \leq 11\bar{M}_1$. Furthermore, we have $\Phi_l < 4$ for $l \geq 3$ by (3). This together with 11-competitiveness, i.e. $\bar{\sigma}_l \geq -2$, yields $\bar{\alpha}_l < 3$ and

$$\bar{M}_{l+1} = (1 + \alpha_l)\bar{M}_l < 4\bar{M}_l \qquad \text{for } l \geq 3.$$

Let $\bar{M} = \bar{M}_l$, where l is chosen to be the first $l \geq 3$ with $\bar{M}_l$ pointing to the right and $\bar{M}_l > 3\varepsilon M$. Thus, either $\bar{M} = \bar{M}_3$ or $\bar{M} = M_4$ and hence $\bar{M} < 100\varepsilon M$, or $l > 4$ and $\bar{M}_{l-2} \leq 3\varepsilon M$, so that $\bar{M}_l \leq 3 \cdot 16\varepsilon M$. In any case we have $3\varepsilon M \leq \bar{M} < 100\varepsilon M$. In particular, there are still unused servers in between M and $\bar{M}$.

Since $l \geq 3$, we have

$$\bar{\Phi} < 4 - \frac{2}{11}$$

(assuming m_0 and hence also $\bar{m}_0$ are large enough). This does not yet imply $\bar{\sigma} < 4$ (which we would like to have in view of (6)). However, as we shall see, $\bar{\alpha} \geq -\frac{1}{2}$ may be assumed. This yields

$$\bar{\sigma} = \bar{\Phi} - 2\bar{\alpha} < 5 - \frac{2}{11}, \tag{7}$$

which will turn out to be good enough for our purposes.

To rule out $\bar{\alpha} < -\frac{1}{2}$, i.e. $\bar{M}_{l+1} < \frac{1}{2}\bar{M}_l$, note that this would imply

$$\begin{aligned} \bar{L}_{l+1} &= 2(\bar{M}_2 + \ldots + \bar{M}_{l+2}) + \bar{M}_{l+1} - (2l+2) \\ &> 2\bar{M}_l + 3\bar{M}_{l+1} + 2\bar{M}_{l+2} \\ &> 4\bar{M}_{l+1} + 3\bar{M}_{l+1} + 4\bar{M}_{l+1}. \end{aligned}$$

So we could force the online algorithm to violate 11-competitiveness in the next step. Thus, we indeed may assume that $\alpha \geq -\frac{1}{2}$ holds, implying (7).

Now we show that, in order to stay $(9+\varepsilon)$-competitive, the algorithm must serve requests $r = M, M+1, \ldots$, etc. for the "first cow" from right, thus extending the current matching M to a point beyond the second cow, as in Figure 4 a).

Indeed, assume to the contrary that the algorithm serves $r = M, M+1, \ldots$ from right and switches back to the left before reaching the "second cow", i.e., it serves some $r \leq \lceil 1.1M \rceil - \bar{M}$ from left. We restrict explicit computations to the case where $r = \lceil 1.1M \rceil - \bar{M}$. (The case $r < \lceil 1.1M \rceil - \bar{M}$ is similar but even easier.)

When the algorithm serves $r = \lceil 1.1M \rceil - \bar{M}$ from left, i.e., from the server at $s = -(1+\alpha)M$, we continue the sequence for the first cow, i.e., we request $r = -(1+\alpha)M, -(1+\alpha)M - 1$, etc. until eventually the algorithm switches back to the current matching $\tilde{M}$ (cf. figure 5).

Figure 5: $\tilde{M} = \lceil 1.1M \rceil + (1+\bar{\alpha})\bar{M} - \bar{M}$

Using $\bar{\alpha} \leq 3$ and $\bar{M} \leq 0.1M$, we find

$$\tilde{M} \leq \lceil 1.1M \rceil + \bar{\alpha}\bar{M} \leq 1.5M.$$

On the other hand, the additional travel length is

$$\Delta L \geq 2(2+\alpha)M + 0.1M$$

So the total travel length would be

$$\begin{aligned} \tilde{L} &= \bar{L} + L + \Delta L \\ &\geq L + \Delta L \geq (13 + 2\alpha - \sigma + 0.1)M > 15M. \end{aligned}$$

(Recall that $\alpha > 1 - \sqrt{\varepsilon}$ and $\sigma < \varepsilon/5$.) So $\tilde{L}/\tilde{M} > 10$, a contradiction.

Hence the first cow is forced to eat up the second in the next step, resulting in a "combined cow" with potential

$$\tilde{\Phi} \leq \frac{M'}{\tilde{M}}\Phi' + \frac{\bar{M}}{\tilde{M}}(\bar{\sigma} - 4) \leq \frac{M'}{\tilde{M}}(\Phi + \varepsilon) + \frac{\bar{M}}{\tilde{M}}(1 - \frac{2}{11}).$$

Now $\Phi > 2 - 2\sqrt{\varepsilon}$ by assumption (otherwise we would have had $\Delta \geq \frac{1}{16}\varepsilon$, cf. Lemma 1). So the upper bound for $\tilde{\Phi}$ is maximized by taking $\bar{M}$ as small as possible. By definition, however, $\bar{M} > 3\varepsilon M$. Since $\Phi < 4 - \frac{2}{9+\varepsilon} < 4 - \frac{1}{5}$, we certainly have $\alpha = (\Phi - \sigma)/2 < (\Phi + \varepsilon)/2 < 2$, so $M' = (1+\alpha)M \leq 3M$, i.e., $\bar{M} > \varepsilon M'$. Hence

$$\tilde{\Phi} \leq \frac{1}{1+\varepsilon}(\Phi + \varepsilon) + \frac{\varepsilon}{1+\varepsilon}(1 - \frac{2}{11})$$

Now $\Phi \geq 2 - 2\sqrt{\varepsilon} \geq 2 - \frac{1}{11}$ yields $\tilde{\Phi} \leq \Phi - \frac{1}{11}\varepsilon$, proving the desired significant decrease in Φ.

Summarizing, we can thus force a decrease of $\Delta \geq \frac{1}{16}\varepsilon$ or $\tilde{\Delta} \geq \frac{1}{11}\varepsilon$ in each step, so that eventually the potential will drop below $2 - 2\sqrt{\varepsilon}$ and then continue to drop further automatically towards $-\infty$, a contradiction. We thus have proved

Theorem 1 *Any ρ-competitive algorithm for matching on a line must have ratio $\rho \geq 9.001$.*

4 Work Functions

In this section we investigate a rather straightforward online matching algorithm and show that it has infinite competitive ratio. The algorithm is based on the concept of *work functions*, which have already been shown to be useful in standard online server problems, cf [3] or [2].

In our context, a work function algorithm can be defined as follows. Assume the online algorithm has already served requests $R = \{r_1, \ldots, r_t\}$, $t \geq 0$, from $S = \{s_1, \ldots, s_t\}$. The size of the corresponding current matching (the optimal matching from S into R) is then called the *work function* of S, denoted by $w_t(S)$. When the new request r_{t+1} arrives, we determine s_{t+1} to be the server that minimizes

$$\gamma \Delta w + d,$$

where $\Delta w = w_{t+1}(S \cup \{s_{t+1}\}) - w_t(S)$ and d is the distance from s_{t+1} to r_{t+1}. The weighting factor $\gamma \geq 0$ can be chosen arbitrarily. The choice $\gamma = 0$ corresponds to the simple greedy strategy serving each new request from the nearest server.

To simplify our analysis, we chose $\gamma = 3$. This results in an online algorithm that asymptotically follows the doubling technique when applied to simple cow sequences:

Figure 6: A simple cow

In the situation indicated in figure 6, choosing s_{t+1} to be the left server s_- would give $\Delta w = 1$ and $d = 1$, so $3\Delta w + d = 4$. For the right server we find $3\Delta w + d < 4$ as soon as the current matching size is roughly 2/3 of the distance between s_+ and the new request.

Though this algorithm performs optimally (with competitive ratio 9) on simple cow sequences, it has infinite competitive ratio in general. To see this, consider k cow sequences next to each other:

Figure 7: k cows

Assuming that the algorithm has already (approximately) spent factor 9 on each of the cow sequences and that there is (at least) one unused server between each of them at positions $s_1, s_2, \ldots, s_k$. A new request at position s_1 will be served from s_1. A second request at s_1 will then be served from s_2 and after that, a request at s_2 will be served from s_3 etc. Finally, a request on s_k will be served from $s_k - 1$, a request there from $s_k - 2$, etc., until finally a request on position (roughly) $s_k - 6M$ will be served from s_0. At this point in time, our current matching looks like indicated in figure 8 and

Figure 8: k concatenated cows

the algorithm has spent (approximately) 15 times the current matching on this type of *concatenated cow sequence*.

It is now straightforward to iterate this argument, placing a number of such concatenated cow sequences next to each other and proving a lower bound

of 21 for the competitive ratio etc. So our algorithm has indeed unbounded competitive ratio.

Other values of γ can be analyzed similarly, so it seems that (standard) work function algorithms are of no help in online matching. Or, to put it differently: Whether to chose the left or right server s_- resp. s_+ for serving a new request should probably be decided by also taking into account the situation outside the interval $[s_-, s_+]$.

References

[1] R. Baeza-Yates, J. Culberson and G. Rawlins, Searching in the plane. *Information and Computation* **106**, 1993, 234–252.

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

[3] E. Koutsoupias and C. Papadimitriou, On the k-server conjecture. *Journal of the ACM* **42**(5), 1995, 971–983.

[4] C. Papadimitriou and M. Yannakakis, Shortest paths without a map. *Theoretical Computer Science* **84**, 1991, 127–150.

[5] K. Pruhs, Personal Communication (2000).