Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 332 (2005) 251 – 264
Theoretical Computer Science
www.elsevier.com/locate/tcs

Online matching on a line

Bernhard Fuchs[a,*,1], Winfried Hochstättler[b], Walter Kern[c]

[a] Center for Applied Computer Science Cologne, Universität zu Köln, Weyertal 80, D-50931 Köln, Germany
[b] Department of Mathematics, BTU Cottbus, Postfach 10 13 44, D-03013 Cottbus, Germany
[c] Department of Applied Mathematics, University of Twente, P.O. Box 217, NL-7500 AE Enschede, Netherlands

Received 1 January 2003; received in revised form 13 September 2004; accepted 11 October 2004
Communicated by A. Fiat

Abstract

Given a set $S \subseteq \mathbb{R}$ of points on the line, we consider the task of matching a sequence $(r_1, r_2, \ldots)$ of requests in $\mathbb{R}$ to points in S. It has been conjectured [Online Algorithms: The State of the Art, Lecture Notes in Computer Science, Vol. 1442, Springer, Berlin, 1998, pp. 268–280] that there exists a 9-competitive online algorithm for this problem, similar to the so-called "cow path" problem [Inform. and Comput. 106 (1993) 234–252]. We disprove this conjecture and show that no online algorithm can achieve a competitive ratio strictly less than 9.001.

Our argument is based on a new proof for the optimality of the competitive ratio 9 for the "cow path" problem.

Keywords: Competitive analysis; Matching; Online algorithms

1. Introduction

We consider a special class of online server problems, where a number of servers (not necessarily finite), located on the real line, is to serve a sequence of requests $r_1, r_2, \ldots, r_k \in \mathbb{R}$. In contrast to classical server problems (cf, e.g. [2,4]), however, each server can serve at most one request. So the optimal offline solution is the minimum cost matching of the

* Corresponding author.
E-mail addresses: bfuchs@zpr.uni-koeln.de (B. Fuchs), hochstaettler@math.tu-cottbus.de (W. Hochstättler), w.kern@math.utwente.nl (W. Kern).
1 Supported by Deutsche Forschungsgemeinschaft, Graduiertenkolleg Scientific Computing, GRK 192/5-02.

doi:10.1016/j.tcs.2004.10.028

requests into the set of server positions s_i. The problem is therefore also known as the *online matching problem on a line* [6]. As an application, consider a Bowling Center with bowling shoes of sizes $s_1, s_2, \ldots$ at its disposal to meet requested shoe sizes $r_1, r_2, \ldots$ of entering players.

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.

For notational convenience, we consider a "universal" instance with infinitely many servers, one at each integer $s \in \mathbb{Z}$. The lower bound on ρ we shall derive is easily extended (cf. Section 4) to the *finite* case, where there is only a finite number of servers given, say, one at each integral $s \in [-N, N]$ for sufficiently large N, and requests $r_1, \ldots, r_k \in \mathbb{R}$ (with $k \leqslant 2N + 1$).

In the next section we will simulate the famous "cow path" problem, which is known to have an optimal online algorithm with competitive ratio of 9 [1], with an instance for the matching problem on a line. In Section 3 we present a new proof for optimality of this competitive ratio. In Section 4 we extend this result to a lower bound of $9+\varepsilon$ for the online matching problem on a line with ε=0.001, contradicting a conjecture presented in [6] that a competitive ratio of 9 can be achieved. Our choice of ε is not optimized but our method does not seem to yield a significantly larger lower bound.

In [6] it is also suggested that generalized work function algorithms might perform well. In Section 5 we show that these algorithms have infinite competitive ratio.

2. The cow path problem

The authors of [6] call the following problem "hide and seek", but more often it is referred to as the "cow path" problem, interpreted as a cow trying to escape from the meadow and looking for a hole in the fence [7]. Mathematically, the fence is represented by the real line and the cow's initial position is the origin. We are seeking for a path visiting each $x \in \mathbb{Z}$ (each possible location of the hole) after traveling a distance of at most $\rho|x|$. Such a path is called a ρ-competitive path (solution) to the (*discrete*) *cow path problem*. Any such path will without loss of generality first lead to $l_1 < 0$, then turn to the right until it reaches $l_2 > 0$, turn again and move to $l_3 < l_1$, and so on. Thus, such a cow path is completely characterized by the sequence of its turning points $l_1, l_2, l_3, \ldots \in \mathbb{Z}$.

The basic difficulty for an online algorithm for the matching problem on the line 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_-.

The following request sequence forces any online algorithm for the matching problem to simulate a "cow path". The first two requests are at $r_1 = r_2 = 0$, and each subsequent request is exactly at the position where a server has just been moved off to serve the previous request. Assume that r_2 is served from $s_2 = -1$. In order to stay ρ-competitive, the online algorithm may first continue to serve a number of requests from left, but must eventually *switch* to serving some request $r = i \leqslant -1$ from right, i.e. from $s = 1$. (Indeed, $|i| \leqslant \rho/2$). It may then continue to serve a number of requests from right, but eventually it will have to switch again, serving some request $r = j \geqslant 1$ from left, etc. Thus the online algorithm for such an instance is characterized by its turning points $l_1, l_2, l_3, \ldots$ which can be interpreted as a cow path.

Proposition 1. *Any ρ-competitive algorithm for online matching on a line yields a ρ-competitive algorithm for the discrete cow problem.*

Proof. Consider a request sequence as described above that stops when $s = x$ is used as a server. Assume that our online algorithm produces a sequence $l_1, l_2, l_3, \ldots, l_k$ with $l_i < 0$ for i odd and $i > 0$ for i even. The constructed online matching then has a cost of $|x| + 2\sum_{i=1}^{k} l_i$, whereas the optimum matching costs $\min\{|x|, |l_k| + 1\}$, since serving $r_2 = 0$ from x resp. $l_k \pm 1$, all the other requests can be matched at no cost. To see this, note, that the request sequence consists of all integers in $[x + 1, l_k]$ resp. $[l_k, x - 1]$ where 0 is requested twice. Obviously, the cost of the online matching equals the cost of a cow path with turning points $l_1, l_2, l_3, \ldots, l_k$. $\square$

This analogy yields a lower bound of $\rho \geqslant 9$ for the competitive ratio of any online algorithm for matching on a line, cf. [1] or Section 3.

For future purposes we, additionally, scale the above sequence and start with $2m_0$ requests at $r = 0, \pm 1, \pm 2, \ldots, \pm(m_0 - 1), 0$. Now the second request at $r = 0$ will be served, say, from $s = -m_0$. We then continue requesting exactly at the positions where a server has just been moved off. We refer to such a request sequence as a *cow sequence* with parameter m_0, started at $r = 0$.

3. Cow sequences

Consider an online algorithm for the matching problem on a line and assume it has already served requests $r_1, \ldots, r_t \in \mathbb{Z}$. 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 2 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. Fig. 1).

In the situation indicated in Fig. 1 we have $M = j, L = 2i + j$ and, assuming that $j > i$, $M^* = i + 1$. In our figures, we indicate unused servers by $\circ$. Note, that always $M_1 = m_0$ and, in terms of turning points $l_1, l_2, \ldots$ of a cow path we have $|M_{i+1}| = |l_i| + 1$ for $i = 1, 2, \ldots$.

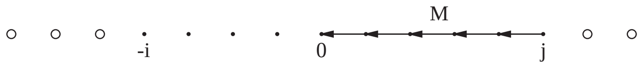

Fig. 1. The current matching M ($m_0 = 1$).

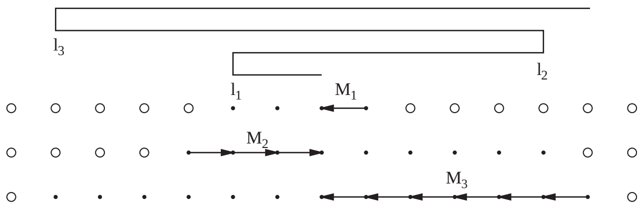

Fig. 2. A cow path and corresponding current matchings M_k.

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 $\rho \geqslant 9$ on cow sequences). When the online algorithm serves a cow sequence, we let $M_k, k \geqslant 1$, denote the current matching immediately after the kth switch (cf. Fig. 2).

Proposition 2. *After the kth switch, when the current matching is M_k, the online algorithm has travelled $L_1 = 2l_1 + m_0 = 2M_2 + M_1 - 2$ if $k = 1$ and*

$$L_k = 2\sum_{i=2}^{k-1} M_i + 3M_k + 2M_{k+1} - 2k, \text{ for } k \geqslant 2. \tag{1}$$

Proof. For $k \geqslant 2$, $L_k = 2\sum_{i=1}^{k} l_k + M_k = 2\left(\sum_{i=2}^{k+1} M_i - 1\right) + M_k$ and the claim follows. □

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 $k \geqslant 2$. 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

$$L_k = 9M_k - 4M_1 - 2k \tag{2}$$

implying

Corollary 3. *The doubling technique is 9-competitive for serving cow sequences.*

To see that 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 kth switch. Let σ_k and α_k be such that

$$L_k = (9 - \sigma_k)M_k \qquad \text{and} \qquad M_{k+1} = (1 + \alpha_k)M_k. \tag{3}$$

Remark 4. The doubling technique would correspond to $\alpha_k = 1,\ k \geqslant 1$. In general only $\alpha_k > -1$ holds by definition, thus, α_k may be negative, and M_k is not guaranteed to be the current optimal assignment for all $k \geqslant 1$. For a 9-competitive algorithm, $\sigma \geqslant 0$ indicates the current "length credit" (relative to the current M) and α can be interpreted as the "credit we have gained by exploring a region of size $(1+\alpha)M$ on the opposite side". In this sense the potential defined below may be interpreted as a kind of "total current credit".

We introduce the *potential*

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

In the following we derive a recursion for Φ_k, showing that any $(9-\varepsilon)$-competitive algorithm would yield $\Phi_k \to -\infty$, contradicting $\sigma \geqslant 0$ and $\alpha > -1$.

Our recursion starts as follows:

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

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 any ρ-competitive algorithm must necessarily produce exponentially growing M_k's in the following sense.

Lemma 5. *Any ρ-competitive algorithm must satisfy*

(1) $M_{k+2\lceil\rho\rceil} \geqslant 2M_k$,

(2) $M_k \leqslant \frac{\rho}{2} M_{k-1}$.

Proof. Assume $M_{k+2\lceil\rho\rceil} < 2M_k$ and consider the situation immediately after the $(k+2\lceil\rho\rceil)$th switch. Then

$$\begin{aligned} L_{k+2\lceil\rho\rceil} &= 2\sum_{i=2}^{k+2\lceil\rho\rceil-1} M_i + 3M_{k+2\lceil\rho\rceil} + 2M_{k+2\lceil\rho\rceil+1} - 2k \\ &\geqslant 2\sum_{i=0}^{\lceil\rho\rceil-1} M_{k+2i} \geqslant 2\sum_{i=0}^{\lceil\rho\rceil-1} M_k \\ &> \lceil\rho\rceil M_{k+2\lceil\rho\rceil}, \end{aligned}$$

contradicting ρ-competitiveness.

By Proposition 2 for $k \geqslant 3$ we have $L_{k-1} \geqslant 3M_{k-1} + 2M_k$ implying the second assertion. □

The first inequality of the previous lemma implies that $\frac{k}{M_k}$ $\left(\text{and even } \sum \frac{k}{M_k}\right)$ can be made arbitrarily small by an appropriately large choice of m_0. The second inequality gives a rough upper bound on Φ_k as follows.

Lemma 6. *For* $k \geqslant 3$

$$\Phi_k < 4 - \frac{2}{\rho}, \tag{4}$$

for m_0 *sufficiently large.*

Proof.

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

Dividing by M_k yields

$$\Phi_k \leqslant 4 - \frac{4}{\rho} + \frac{2k}{M_k} < 4 - \frac{2}{\rho}$$

for m_0 sufficiently large. □

Next we derive the recursion for Φ_k.

Lemma 7.

$$\Phi_{k+1} = \Phi_k - \Delta_k + \frac{2}{M_{k+1}} \quad \textit{with} \quad \Delta_k = \frac{\alpha_k\sigma_k + 2(1 - \alpha_k)^2}{1 + \alpha_k}. \tag{5}$$

Proof. We compute from Proposition 2 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 $M_{k+1} = (1 + \alpha_k)M_k$, $M_{k+2} = (1 + \alpha_{k+1})(1 + \alpha_k)M_k$ and dividing by 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 $1 + \alpha_k$ yields the recursion. □

Remark 8. 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 a $(9 - \varepsilon)$-competitive algorithm for serving cow sequences (and hence, a fortiori, for matching on a line) cannot exist. Such an algorithm would maintain $\sigma_k \geqslant \varepsilon > 0$. This implies

Lemma 9. *If $\sigma_k \geqslant 0$ we have $\Delta_k \geqslant \frac{1}{3}\sigma_k$. If, furthermore, $\sigma_k \geqslant \varepsilon > 0$ for all k then*

$$\Delta_k \geqslant \tfrac{1}{3}\varepsilon > 0 \text{ for all } k.$$

Proof.

$$\begin{aligned}\Delta_k - \frac{1}{3}\sigma_k &= \frac{\alpha_k\sigma_k + 2(1-\alpha_k)^2}{1+\alpha_k} - \frac{1}{3}\sigma_k \\ &= \frac{\frac{1}{3}\alpha_k\sigma_k + \frac{1}{3}\sigma_k(\alpha_k - 1) + 2(1-\alpha_k)^2}{1+\alpha_k}.\end{aligned}$$

Since the minimum of the denominator of the fraction in the last line, for given $\sigma_k \geqslant 0$, is attained at $\alpha_k = 1 - \frac{1}{6}\sigma_k$, the claim follows. □

So the update $\Phi_{k+1} = \Phi_k - \Delta_k$, and, according to Remark 8, $\Phi_{k+1} = \Phi_k - \Delta_k + \frac{2}{M_k+1}$, would yield $\lim_{k\to\infty} \Phi_k \to -\infty$, whereas $\Phi_k = \sigma_k + 2\alpha_k \geqslant \varepsilon + 2(-1)$ must hold, a contradiction. Our approach also reveals that any 9-competitive algorithm must asymptotically follow the doubling technique when serving a cow sequence.

Theorem 10. *Any online algorithm for matching on a line that is 9-competitive for cow sequences produces σ_k, α_k with $\lim_{k\to\infty} \sigma_k = 0$ and $\lim_{k\to\infty} \alpha_k = 1$.*

Proof. By Lemma 9 $\sigma_k \geqslant 0$ for all k implies that $\Delta_k \geqslant 0$ in Lemma 7 and further, $\sum_{j \geqslant k} \Delta_j$ 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 11. *For a $(9+\varepsilon)$-competitive algorithm serving a cow sequence with m_0 sufficiently large and $0 \leqslant \varepsilon \leqslant \frac{1}{4}$ we have in iteration $k \geqslant 3$*
(1) $\Delta_k \geqslant -\varepsilon$,
(2) $\alpha_k \leqslant 1 - \frac{3}{4}\sqrt{\varepsilon} \Rightarrow \Delta_k \geqslant \frac{1}{16\varepsilon}$,
(3) $\Phi_k \leqslant 2 - 2\sqrt{\varepsilon} \Rightarrow \Delta_k \geqslant \frac{1}{16}\varepsilon$.

Proof. By Lemma 6 we have for $k \geqslant 3 : \Phi_k < 4 - \frac{2}{9+\varepsilon} \leqslant 4 - \frac{1}{5}$. Thus, in case $-1 < \alpha < 0$ we get

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

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

By Lemma 7, $\Delta_k \geqslant \frac{\alpha_k}{\alpha_k+1}\sigma_k \geqslant \frac{\alpha_k}{\alpha_k+1}(-\varepsilon) \geqslant -\varepsilon$. This proves 1.

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

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

which proves (2).

Finally, $0 \leqslant \varepsilon \leqslant \frac{1}{4}$ yields $\varepsilon \leqslant \frac{\sqrt{\varepsilon}}{2}$. Thus, $\Phi_k \leqslant 2 - 2\sqrt{\varepsilon}$ and $\sigma_k \geqslant -\varepsilon$ implies $\alpha_k \leqslant 1 - \frac{3}{4}\sqrt{\varepsilon}$. □

4. More cows

The basic idea for proving a lower bound $\rho \geqslant 9 + \varepsilon$ for online matching is to run two (or more) cow sequences. Assume, we have two "cows" with current matchings $M = M_k$ and $\bar{M} = \bar{M}_l$, directed away from each other, as indicated in Fig. 3. We will omit indices if all parameters in question are indexed by k.

Assume that the first cow sequence is continued, i.e. $r = M, M + 1$, etc. are requested. Furthermore, assume the online algorithm serves all these requests from right, thus extending M to some point "beyond the second cow" (cf. Fig. 4(a)) until it switches back to $M' = M_{k+1}$ (cf. Fig. 4(b)).

This results in a *combined cow* (cf. Fig. 4(b)) in the sense that, when the request sequence is continued with $r = -M', -M' - 1, \ldots$, the online algorithm behaves as 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', where Φ' is the same as the potential of the first cow immediately after switching, disregarding the current matching $\bar{M}$ of the second cow. In particular, Lemmas 11(1) and 7 imply

$$\Phi' \leqslant \Phi + \varepsilon + \frac{2}{M'}. \tag{6}$$

Furthermore, the "combined cow" has scanned the same area as the "first cow", i.e., we have the *total range equality*

$$(2 + \alpha')M' = (2 + \tilde{\alpha})\tilde{M}. \tag{7}$$

The effect of "eating up the second cow" is that, under certain circumstances (cf. below), the potential $\tilde{\Phi}$ of the combined cow is smaller than Φ'.

The parameters $\tilde{\alpha}, \tilde{\sigma}$, etc. of the combined cow are easily computed from the parameters $\bar{\alpha}, \bar{\sigma}$, etc. of the second cow and the parameters α', σ', etc. of the first cow (after the next switch, disregarding the second cow).

Lemma 12. *The new parameters $\tilde{M}, \tilde{L}, \tilde{\alpha}, \tilde{\sigma}, \tilde{\Phi}$ satisfy*

(1) $\tilde{\sigma}\tilde{M} = \sigma' M' + \bar{\sigma}\bar{M}$,

(2) $\tilde{\alpha}\tilde{M} = \alpha' M' - 2\bar{M}$,

(3) $\tilde{\Phi} = \frac{M'}{\tilde{M}}\Phi' + \frac{\bar{M}}{\tilde{M}}(\bar{\sigma} - 4)$.

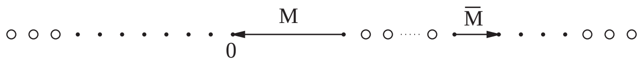

Fig. 3. Two cows in opposition.

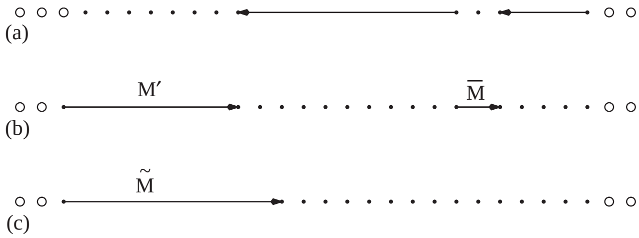

Fig. 4. Combining two cows.

Proof. Clearly, $\tilde{L} = \bar{L} + L'$ and thus

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

implying the first equation. The second assertion follows directly from the total range equality (7).

The combined potential is now easily computed

$$\tilde{\Phi} = \tilde{\sigma} + 2\tilde{\alpha} = \frac{M'}{\tilde{M}}(\sigma' + 2\alpha') + \frac{\bar{M}}{\tilde{M}}(\bar{\sigma} - 4). \qquad \square$$

In particular, $\tilde{\Phi}$ is significantly less than Φ', for example, when $\bar{\sigma} < 4$. In view of (6), we may even expect that $\tilde{\Phi}$ is significantly smaller than Φ.

This is the basic idea of our approach: We run a cow sequence as long as the potential decreases significantly, say $\Delta \geqslant \frac{\varepsilon}{16}$. 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. The potential will, thus, eventually drop below $2 - 2\sqrt{\varepsilon}$. From this point on, the potential decreases automatically (cf. Lemma 11), 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 $\Delta \geqslant \frac{\varepsilon}{16}$, we continue the sequence. Eventually, since $\Phi > -\varepsilon - 2$, $\Delta < \frac{\varepsilon}{16}$ must occur, implying

$$\sigma < \frac{3\varepsilon}{16} \leqslant \frac{\varepsilon}{5} \qquad \text{and} \qquad \alpha > 1 - \frac{3}{4}\sqrt{\varepsilon} \geqslant 1 - \sqrt{\varepsilon}$$

by Lemmas 9 and 11.

Assume w.l.o.g. that the current matching $M = M_k$ points to the left as in Fig. 3. We then start a second cow at $\bar{r} = \lceil 1.1M \rceil$ with $\bar{m}_0 = \lceil \varepsilon M \rceil$. The total length credit that we inherit from the first cow is $(\sigma+\varepsilon)M \leqslant \frac{6}{5}\varepsilon M$. We compute

$$\begin{aligned}(9-\sigma)M + \bar{L} &\leqslant (9+\varepsilon)(M+\bar{M})\\ \Rightarrow \bar{L} &\leqslant \frac{6}{5}\varepsilon M + (9+\varepsilon)\bar{M}\\ &\leqslant \frac{6}{5}\bar{m}_0 + (9+\varepsilon)\bar{M} \leqslant \left(9+\frac{6}{5}+\varepsilon\right)\bar{M}.\end{aligned}$$

So the second cow is certainly bound to be 11-competitive. Assume it produces current matchings $\bar{M}_k$. Then

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

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

$$\bar{\alpha}_l < 3 \text{ and } \bar{M}_{l+1} = (1+\bar{\alpha}_l)\bar{M}_l < 4\bar{M}_l \qquad \text{for } l \geqslant 3. \tag{8}$$

Lemma 13. *Let $\bar{M} = \bar{M}_l$, where L is chosen to be the first $l \geqslant 3$ with $\bar{M}_l$ pointing to the right and $\bar{M}_l > 3\varepsilon M$. Then*

$$3\varepsilon M \leqslant \bar{M} < 100\varepsilon M. \tag{9}$$

Thus, there still are unused servers in between M and $\bar{M}$.

Proof. Either $\bar{M} = \bar{M}_3$ or $\bar{M} = M_4$ and hence $\bar{M} < 100\varepsilon M$, or $l > 4$ and $\bar{M}_{l-2} \leqslant 3\varepsilon M$, so that $\bar{M}_l \leqslant 3 \cdot 16\varepsilon M$. □

Since $l \geqslant 3$, we have

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

(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 Lemma 12). However, the estimate below will turn out to be good enough for our purposes.

Lemma 14.

$$\bar{\sigma} < 5 - \tfrac{2}{11}. \tag{10}$$

Proof. First we show $\bar{\alpha} \geqslant -\frac{1}{2}$. For $\bar{\alpha} < -\frac{1}{2}$, i.e. $\bar{M}_{l+1} < \frac{1}{2}\bar{M}_l$, would imply

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

So we could force the online algorithm to violate 11-competitiveness in the next step. Thus

$$\bar{\sigma} = \bar{\Phi} - 2\bar{\alpha} < 5 - \tfrac{2}{11}. \qquad \square$$

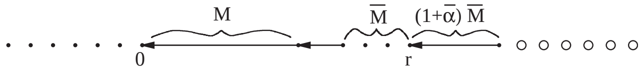

Fig. 5. $\tilde{M} = \lceil 1.1M \rceil + (1+\bar{\alpha})\bar{M} - \bar{M}$.

Lemma 15. *In order to stay* $(9+\varepsilon)$*-competitive, an online algorithm must serve requests* $r = M, M+1, \ldots$, *etc. for the "first cow" from the right, thus extending the current matching* M *to a point beyond the second cow, as in Fig.* 4(a).

Proof. Assume to the contrary that the algorithm serves $r = M, M+1, \ldots$ from the right and switches back to the left before reaching the "second cow", i.e. it serves some $r \leqslant \lceil 1.1M \rceil - \bar{M}$ from the 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 the 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. Fig. 5).

Using $\bar{\alpha} \leqslant 3$ from (8) and $\bar{M} \leqslant 0.1M$, we find

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

On the other hand, the additional (after having reached the situation in Lemma 13) travel length is

$$\Delta L \geqslant 2((1+\alpha)M + M) + (r - M) \geqslant 2(2+\alpha)M + 0.1M.$$

So the total travel length would be

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

(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

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

Now $\Phi > 2 - 2\sqrt{\varepsilon}$ by assumption (otherwise we would have had $\Delta \geqslant \frac{1}{16}\varepsilon$, cf. Lemma 11). 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 (cf. Lemma 6) $\Phi < 4 - \frac{2}{9+\varepsilon} < 4 - \frac{1}{5}$, we certainly have $\alpha = (\Phi - \sigma)/2 \leqslant (\Phi + \varepsilon)/2 < 2$, so $M' = (1+\alpha)M \leqslant 3M$, i.e. $\bar{M} > \varepsilon M'$. Hence, by Lemma 12, (6) and Lemma 14

$$\tilde{\Phi} \leqslant \frac{1}{1+\varepsilon}(\Phi + \varepsilon) + \frac{\varepsilon}{1+\varepsilon}\left(1 - \frac{2}{11}\right).$$

Now, if still $\tilde{\Phi} \geqslant 2 - 2\sqrt{\varepsilon} \geqslant 2 - \frac{1}{11}$ we compute

$$\tilde{\Phi} \leqslant \Phi + 2\varepsilon - \frac{2}{11}\varepsilon - \varepsilon\tilde{\Phi} \leqslant \Phi - \frac{1}{11}\varepsilon,$$

proving the desired significant decrease in potential.

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

Theorem 16. *Any ρ-competitive algorithm for online matching on a line must have ratio $\rho \geqslant 9.001$.*

More precisely, our analysis reveals that Φ drops from $\Phi \leqslant 4$ to $\Phi < -1$ in $O(\varepsilon^{-1})$ switches of the "first" (combined) cow. Using the second inequality in Lemma 5, we easily derive a finite variant of Theorem 16, where servers are located at integral positions in $[-N, N]$ for sufficiently large N and requests $r_1, \ldots, r_k$ ($k \leqslant 2N + 1$).

5. 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. [4] or [2] and have been suggested as good candidates for online algorithms for the matching problem on a line [6].

We will merely restrict to an outline of the construction, as it is easy but tedious to figure out the details. Furthermore, Koutsoupias and Nanavati [3] have, independently, analyzed work functions in more detail. Presenting an easier, but (like ours) hierarchically structured example, they show that the competitive ratio of work function algorithms is $\Omega(\log n)$ and $O(n)$.

In our context, a work function algorithm can be defined as follows. Assume the online algorithm has already served requests $R = \{r_1, \ldots, r_t\}$, $t \geqslant 0$, 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 $\gamma \geqslant 0$ 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.

In the situation indicated in Fig. 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 $\frac{2}{3}$ of the distance between s_+ and the new request.

Fig. 6. A simple cow.

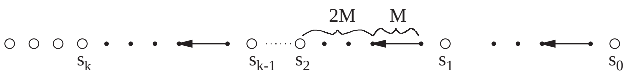

Fig. 7. *k* cows.

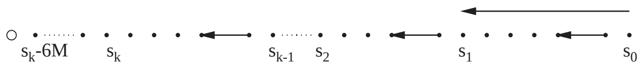

Fig. 8. *k* concatenated cows.

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 as in Fig. 7.

Assuming that the algorithm has already (approximately) spent factor 9 on each of the cow sequences and that there is exactly 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 face work functions of $3(M + 1) + 3M + 1$ for s_2 and $3(3M + 1) + 3M + 1$ for s_0 and thus will then be served from s_2. 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 the one indicated in Fig. 8 and the algorithm has spent (approximately) $9kM + 3kM + 3kM$ which is 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, G. Rawlins, Searching in the plane, Inform. Comput. 106 (1993) 234–252.
[2] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambride University Press, Cambridge, 1998.
[3] E. Koutsoupias, A. Nanavati, The online matching problem on a line, in: K. Jansen, R. Solis-Oba, (Eds.), Approximation and Online Algorithms, First International Workshop, WAOA 2003, Lecture Notes in Computer Science, Vol. 2909, Springer, Berlin, 2004, pp. 179–191.
[4] E. Koutsoupias, C. Papadimitriou, On the *k*-server conjecture, J. ACM 42 (5) (1995) 971–983.
[5] C. Papadimitriou, M. Yannakakis, Shortest paths without a map, Theoret. Comput. Sci. 84 (1991) 127–150.

[6] K. Pruhs, B. Kalyanasundaram, Online network optimization problems, in: A. Fiat, G. Woeginger (Eds.), Online Algorithms: The State of the Art, Lecture Notes in Computer science, Vol. 1442, Springer, Berlin, 1998, pp. 268–280.
[7] G. Woeginger, Personal communication.