Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Note

# Logarithmic price of buffer downscaling on line metrics

Marcin Bienkowski [a,*], Martin Böhm [b], Łukasz Jeż [a], Paweł Laskoś-Grabowski [a,c,d], Jan Marcinkowski [a], Jiří Sgall [b], Aleksandra Spyra [a], Pavel Veselý [b]

[a] Institute of Computer Science, University of Wroclaw, Poland
[b] Computer Science Institute, Charles University, Czech Republic
[c] Electron (Chaddenwych Services Ltd.), London, UK
[d] Institute of Theoretical Physics, University of Wroclaw, Poland

A R T I C L E I N F O

*Article history:*
Received 30 November 2016
Received in revised form 22 July 2017
Accepted 12 October 2017
Available online xxxx
Communicated by G.F. Italiano

*Keywords:*
Reordering buffer
Competitive analysis

A B S T R A C T

We consider the reordering buffer problem on a line consisting of $n$ equidistant points. We show that, for any constant $\delta$, an (offline) algorithm that has a buffer $(1-\delta)\cdot k$ performs worse by a factor of $\Omega(\log n)$ than an offline algorithm with buffer $k$. In particular, this demonstrates that the $O(\log n)$-competitive online algorithm MovingPartition by Gamzu and Segev (2009) [9] is essentially optimal against any offline algorithm with a slightly larger buffer.

## 1. Introduction

In the reordering buffer problem, requests arrive in an online fashion at the points of a metric space and have to be served by an algorithm. An algorithm has a single server kept at a point of the metric space and is equipped with a finite buffer, whose capacity is denoted by $k$. The buffer is used to give the algorithm a possibility of serving requests in a different order. Namely, at any time, there can be at most $k$ unprocessed requests, and once their number is exactly $k$, the algorithm has to move its server to the position of a pending request of its choice. The request is then considered served and removed from the buffer. The goal is to minimize the total distance traveled by the server.

The problem was coined by Räcke et al. [1] and the currently best algorithm for general metric spaces is a randomized $O(\log n \cdot \log k)$-competitive strategy [2,3], where $n$ is the number of points in a metric space. Most of the research focused however on specific metric spaces, such as uniform metrics [1,4,5], stars [6–8] or lines of $n$ equidistant points [9,10].

### 1.1. Two-stage approach

One of the straightforward ways to attack the problem is by a two-stage approach: (i) compare an online algorithm to an optimal offline algorithm that has a smaller buffer $h < k$ and (ii) bound the ratio between optimal offline algorithms equipped with buffers $h$ and $k$. Such an approach was successfully executed by Englert and Westermann [5] for uniform metrics. They constructed an online algorithm Map (with a buffer $k$) that is 4-competitive against Opt$(k/4)$, where Opt$(s)$

* Corresponding author.
*E-mail address:* marcin.bienkowski@cs.uni.wroc.pl (M. Bienkowski).

https://doi.org/10.1016/j.tcs.2017.10.008
0304-3975/© 2017 Elsevier B.V. All rights reserved.

denotes the optimal offline algorithm with buffer $s$. Subsequently, they showed that on any instance the costs of $\textsc{Opt}(k/4)$ and $\textsc{Opt}(k)$ can differ by a factor of at most $O(\log k)$. This implies that $\textsc{Map}$ is $O(4 \cdot \log k) = O(\log k)$-competitive.

As shown by Aboud [11], for the uniform metric space the relation between $\textsc{Opt}(k/4)$ and $\textsc{Opt}(k)$ cannot be asymptotically improved, which excludes the option of beating the ratio $O(\log k)$ by the described two-stage process. Note that this does not rule out the possibility of decreasing the ratio by another approach. Indeed, subsequent works gave improved results, essentially resolving the uniform metric case for deterministic algorithms: Adamaszek et al. [6] gave an $O(\sqrt{\log k})$-competitive deterministic strategy and showed that the competitive ratio of every deterministic algorithm is $\Omega(\sqrt{\log k / \log\log k})$.

### 1.2. Two-stage approach for line metrics (our result)

Arguably, the most uncharted territory is the line metric, or more specifically, a line graph consisting of $n$ equidistant sites. There, the lower bound on the competitive ratio is only 2.154 [9]. On the other hand, the best known strategy is an $O(\log n)$-competitive algorithm MovingPartition by Gamzu and Segev [9]. Note that achieving an upper bound of $O(k)$ is possible for any metric space [5], and hence the competitive ratio for line metrics is $O(\min\{k, \log n\})$. In this paper, we show that it is not possible to improve this bound by the two-stage approach described above, by proving the following result.

**Theorem 1.** *Fix a line consisting of $n$ equidistant sites, any $k$, and any constant $\delta \in (0, 1)$. There exists an input sequence on which the ratio between the costs of $\textsc{Opt}(k)$ and $\textsc{Opt}((1-\delta) \cdot k)$ is at least $\Omega(\min\{k, \log n\})$.*

Our result has additional consequences for the two known online algorithms for the reordering buffer problem: MovingPartition [9] and Pay [3].

A straightforward modification of the analysis in [9] shows that MovingPartition is in fact $O((h/k) \cdot \log n)$-competitive against an optimal offline algorithm that has a larger buffer $h > k$. Our result implies that, assuming $\log n = O(k)$, for $h = k \cdot (1 + \varepsilon)$ and a fixed $\varepsilon > 0$, this ratio is asymptotically optimal.

Similarly, algorithm Pay [3] achieves a competitive ratio of $O((h/k) \cdot (\log D + \log k))$ on trees with hop-diameter $D$, and hence it is $O((h/k) \cdot (\log n + \log k))$-competitive when applied to a line metric. Therefore, Pay is asymptotically optimal for line metrics when $n = O(k)$ and $h = k \cdot (1 + \varepsilon)$ for a fixed $\varepsilon > 0$.

## 2. Main construction

The main technical contribution of this paper is to show Theorem 1 for the case when the line consists of exactly $n = 2^k + 1$ sites. For such a setting, the key lemma given below yields the cost separation of $\Omega(k) = \Omega(\log n)$. In the next section, we show how request grouping extends this result from the case of $n = 2^k + 1$ sites to arbitrary values of $k$ and $n$.

**Lemma 2.** *Fix two integers $0 < \ell' < \ell$ and a constant $\varepsilon$ such that $\ell' = (1-\varepsilon) \cdot \ell$. Assume that the line consists of $2^\ell + 1$ equidistant sites. There exists an input sequence on which the ratio between the costs of $\textsc{Opt}(\ell)$ and $\textsc{Opt}(\ell')$ is at least $\ell \cdot \varepsilon^2 \cdot (1+\varepsilon)^{-1} \cdot \log_2^{-1}(1 + 1/\varepsilon)$.*

We number the sites from 0 to $2^\ell$. For simplicity of the description, we associate times with request arrivals: all requests arrive only during a *step* between two consecutive integer times. For a single step, we will explicitly specify the positions of the requests arriving in this step but not their time ordering, which is irrelevant. (If needed, one may assume that they arrive in the increasing site numbering order.) The input sequence consists of *phases*, one of which we describe below. Then, any subsequent phase is "mirrored" with respect to the previous one, i.e., the description is identical except for reversing the site numbering.

The phase consists of $2^\ell$ steps and it begins at time 0. Each request is either *regular* or *auxiliary*. Each regular request has a *rank* $i$ from 0 to $\ell - 1$ and their positions are defined by the recursive construction of *blocks* described below.

For any integers $q \in \{1, \dots, \ell\}$ and $s \in \{0, \dots, 2^{\ell-q} - 1\}$, a *$(q, s)$-block* is a square portion of space–time, consisting of all times in $(2^q \cdot s, 2^q \cdot (s+1)]$ and all sites in $(2^q \cdot s, 2^q \cdot (s+1)]$, where $q$ is called the *rank* of the block. For instance, the $(\ell, 0)$-block is the whole phase. Moreover, $(q, s)$-block contains two *sub-blocks*: the $(q-1, 2s)$-block and the $(q-1, 2s+1)$-block, see Fig. 1. In each $(q, s)$-block a unique request of rank $q - 1$ arrives at site $2^q \cdot (s+1)$ in the step following time $2^q \cdot s$; note that this request is not contained in any of its two sub-blocks. Graphically, it is near the top left corner of the block, and it is depicted as a disk in Fig. 1.

In turn, auxiliary requests are grouped in sets of $\ell + 1$ called *anchors* (squares in Fig. 1). For all $j$ from 0 to $2^\ell - 1$, the $j$-th anchor arrives at site $j$ in the step following time $j$.

Throughout the paper, any relative directions in the text pertain to this figure, e.g., "up" and "right" mean "with growing site numbers" and "forward in time", respectively.

### 2.1. Upper bound on the cost of the optimal algorithm with a larger buffer

Let the *basic trajectory* (depicted as a thick line in Fig. 1) be the movement of a server that, for all $j$ from 0 to $2^\ell - 1$, remains at site $j$ throughout the step between the integer times $j$ and $j + 1$, and moves directly from $j$ to $j + 1$ at the