

# Improved Online Algorithms for the Sorting Buffer Problem on Line Metrics

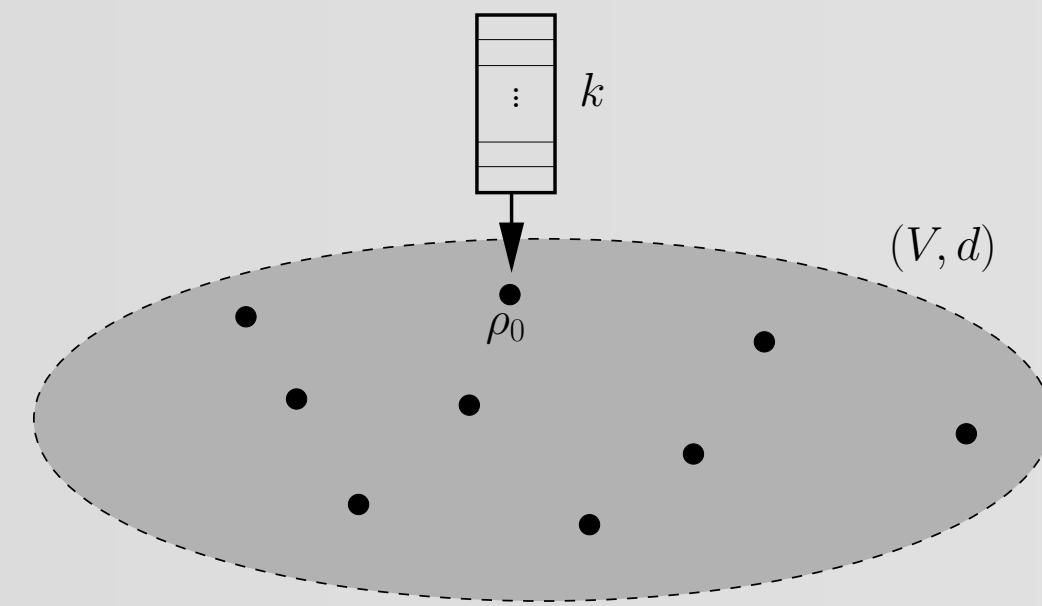


Iftah Gamzu and Danny Segev, Tel Aviv University, Israel

## PROBLEM

An instance of the *sorting buffer* problem consists of:

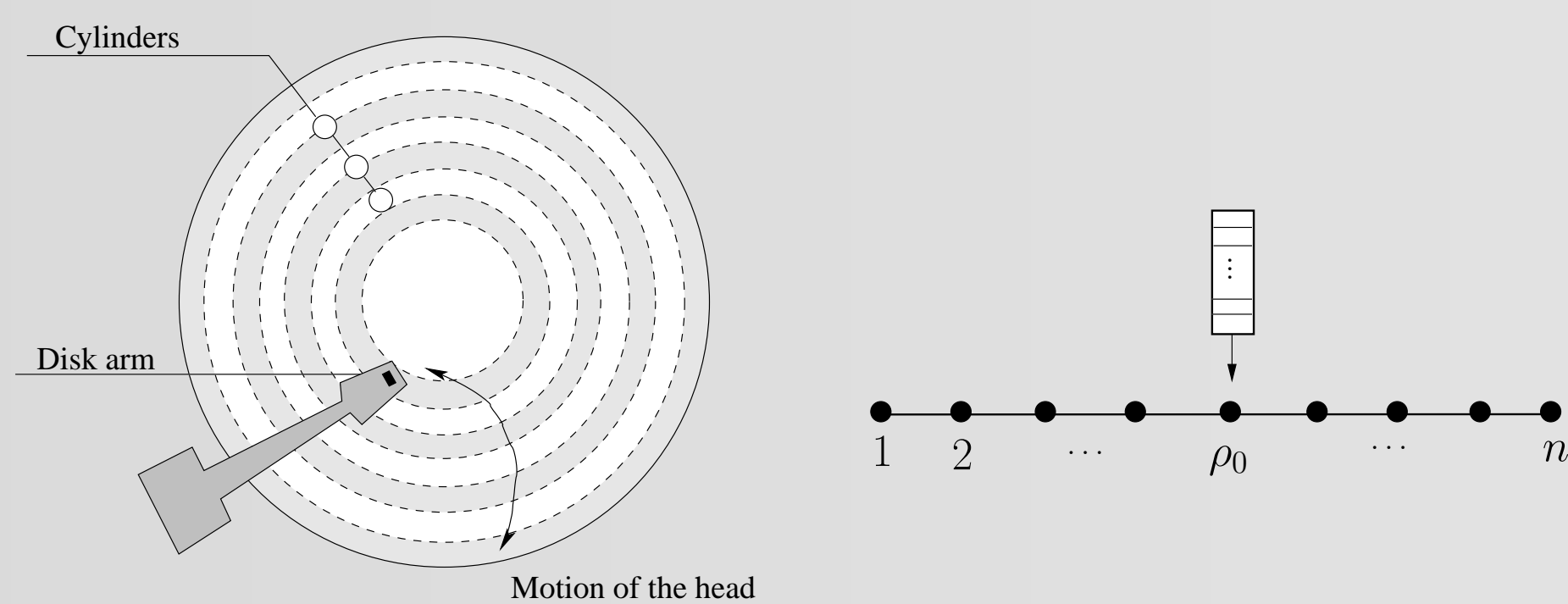
- A *metric space*  $(V, d)$ .
- A *server*, initially positioned at  $\rho_0 \in V$ .
- A *buffer*, capable of holding  $k$  requests.
- An *online sequence*  $\sigma = \langle \sigma_1, \sigma_2, \dots \rangle$  of requests, each of which corresponds to a point in  $V$ .



The objective is to serve all the requests in a way that *minimizes the total distance* traveled by the server.

## MOTIVATION

Consider an *evenly-spaced line metric*, namely, a metric  $(V, d)$  such that  $V = \{1, \dots, n\}$  and  $d(p, q) = |p - q|$ .



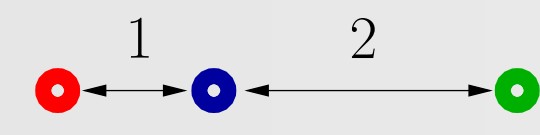
This setting captures a fundamental problem in the design of storage systems known as the *disk arm scheduling* problem [4, 5]. This problem revolves around the development and implementation of efficient buffer-based scheduling policies for disks. The disk's cylinders are modeled by the set of points on the real line, the disk arm corresponds to the server, and the buffer used to reorder read/write requests corresponds to the sorting buffer.

## OUR RESULTS

- We present a *deterministic*  $\Theta(\log n)$ -competitive online algorithm for the sorting buffer problem on an  $n$ -point *evenly-spaced line metric*. This result improves on a *randomized* algorithm proposed by Khandekar and Pandit [2], which obtains an expected competitive ratio of  $O(\log^2 n)$ . It also refutes their conjecture, stating that a deterministic strategy is unlikely to obtain a non-trivial competitive ratio.
- We study the sorting buffer problem on a *continuous line metric*, and utilize the above-mentioned algorithm as a subroutine to devise a deterministic online algorithm that achieves a competitive ratio of  $O(\log N \log \log N)$ , where  $N$  denotes the length of the input sequence.
- We establish the first non-trivial lower bound for the evenly-spaced line case, by proving that the competitive ratio of any deterministic algorithm is at least  $(2 + \sqrt{3})/\sqrt{3} \approx 2.154$ .

## WHY NATURAL STRATEGIES FAIL?

Consider an instance that consists of the 3-point metric schematically described below, a server initially positioned at  $\bullet$ , and a buffer of size 10.



The *first-in-first-out* (FIFO) strategy, which serves the requests in the exact same order by which they arrive, fails as there are cases in which the server travels too far too often (for example, the online sequence  $\sigma = \langle (\bullet, \bullet)^{10} \rangle$  admits an optimal schedule of distance 1, whereas the FIFO schedule has a distance of 19).

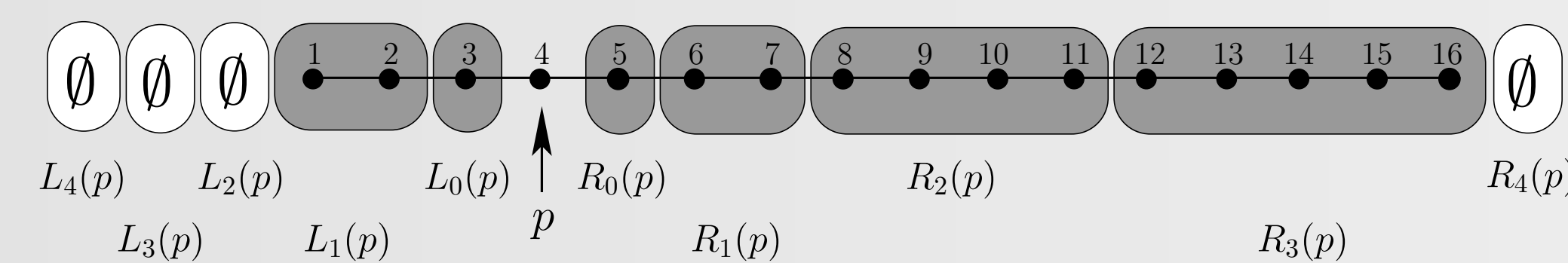
The *nearest-first* (NF) heuristic, which serves the nearest pending request with respect to the server, fails as there are cases in which many requests are kept pending, thus reducing the effective size of the buffer (for instance, the online sequence  $\sigma = \langle \bullet^9, (\bullet, \bullet)^{10} \rangle$  admits an optimal schedule of distance 6, whereas the NF schedule has a distance of 23).

## DOUBLING PARTITIONS

A *doubling partition* with respect to a point  $p \in V$ , henceforth denoted by  $DP(p)$ , is a pairwise-disjoint partition of  $V \setminus \{p\}$  into  $m = 2 \log n$  points sets  $L_0(p), \dots, L_{\log n}(p), R_0(p), \dots, R_{\log n}(p)$ , where

$$L_i(p) = \{q < p : 2^i \leq d(q, p) < 2^{i+1}\} \text{ and}$$

$$R_i(p) = \{q > p : 2^i \leq d(q, p) < 2^{i+1}\}.$$



The figure shows a concrete doubling partition in an evenly-spaced 16-point line metric with respect to the point  $p = 4$ . Note that empty doubling partition sets are marked with  $\emptyset$ .

## ALGORITHM FOR EVENLY-SPACED LINE METRICS

Our algorithm addresses the shortcomings of the natural strategies by striking a balance between clearing the buffer vigorously, which results in travelling too far too often, and keeping many requests pending, which reduces the effective size of the buffer. It is deterministic, and achieves a competitive ratio of  $\Theta(\log n)$ .

The algorithm works in phases, each of which is logically built from three steps:

### (1) Initialization step:

- Let  $p$  be the current position of the server.
- Associate a unique  $k/m$ -sized sub-buffer with each of the  $m$  points sets in  $DP(p)$ .

### (2) Accumulation step:

- Store arriving requests in their associated sub-buffer.
- This step ends when one of the *sub-buffers becomes full* or when the *sequence of requests ends*.

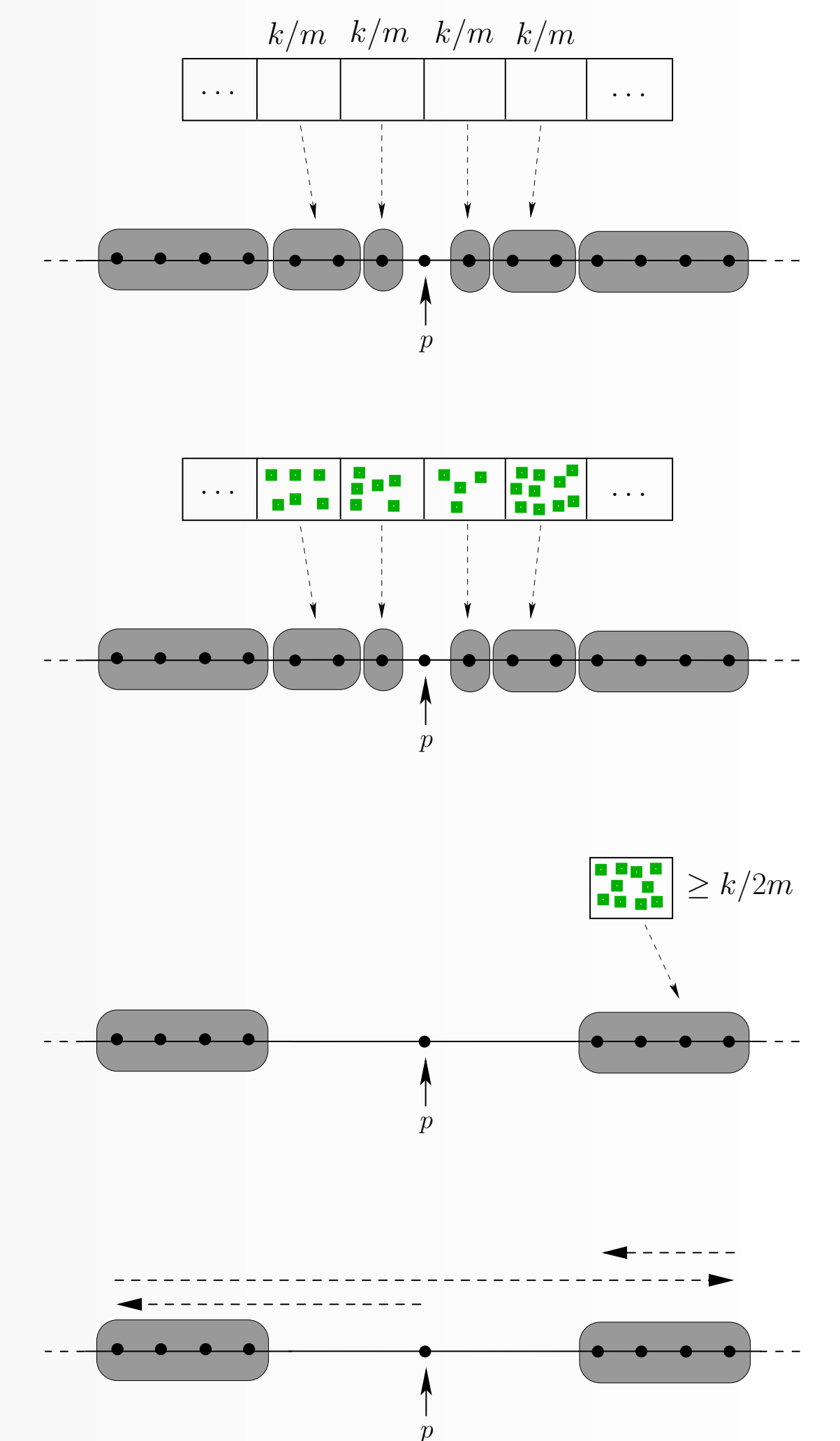
### (3) Clearance step:

Case I: If one of the sub-buffers is full...

- Let  $t$  be the *maximal index* for which  $R_t(p)$  or  $L_t(p)$  is *half-full*, that is, the associated buffer contains at least  $k/2m$  pending requests (wlog, let  $R_t(p)$  be that set).
- Move the server and clear all pending requests along the way:  $p \rightsquigarrow$  leftmost point of  $L_t(p) \rightsquigarrow$  rightmost point of  $R_t(p) \rightsquigarrow$  leftmost point of  $R_t(p)$ .

Case II: If the sequence of requests ends...

- Clear all pending requests by moving the server:  $p \rightsquigarrow$  leftmost point in the buffer  $\rightsquigarrow$  rightmost point in the buffer.



## DISCUSSION

The sorting buffer problem is motivated by numerous real-life applications in networking, file server management, computer graphics, and even in the automotive industry [3, 1]. However, there are only a handful of results addressing this problem, and there are many challenging open research questions:

- One natural question is to *close the gap* between 2.15 and  $O(\log n)$  for the evenly-spaced line case. Alternatively, it would be interesting to obtain a performance guarantee of  $o(k)$ .
- Another intriguing open question is to understand the *offline scenario*, in which the input sequence  $\sigma$  is known in advance.
- Finally, it would be interesting to attain any non-trivial result for *general metrics*.

## REFERENCES

- [1] M. Englert and M. Westermann. Reordering buffer management for non-uniform cost models. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, pages 627–638, 2005.
- [2] R. Khandekar and V. Pandit. Online sorting buffers on line. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 584–595, 2006.
- [3] H. Räcke, C. Sohler, and M. Westermann. Online scheduling for sorting buffers. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 820–832, 2002.
- [4] A. Silberschatz, P. B. Galvin, and G. Gagne. *Applied operating system concepts*. John Wiley and Sons, Inc., first edition, 2000. Disk scheduling is discussed in Section 13.2.
- [5] A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall PTR, second edition, 2001. Disk scheduling is discussed in Section 5.4.

## FUNDING

The first author was supported in part by the German-Israeli Foundation and by the Israel Science Foundation.

