

Improved Online Algorithms for the Sorting Buffer Problem

Danny Segev

School of Mathematical Sciences, Tel-Aviv University

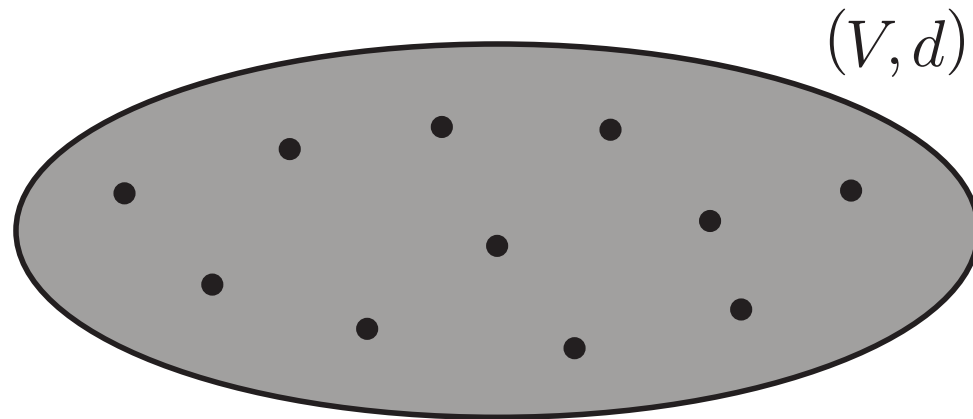
Joint work with *Iftah Gamzu*, School of Computer Science, TAU

Outline

- The sorting buffer problem
- Previous work
- Evenly-spaced line metrics
- Continuous line metrics
- A deterministic lower bound
- Concluding remarks

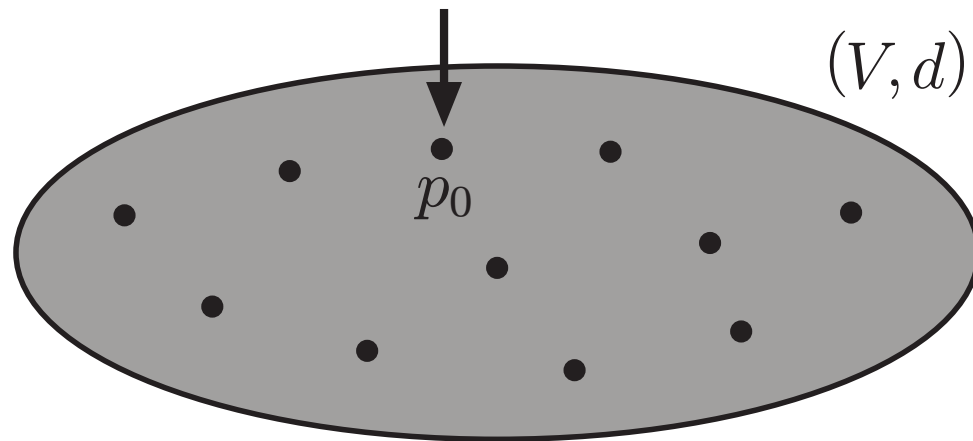
The Sorting Buffer Problem

Input: ■ A metric space (V, d) .



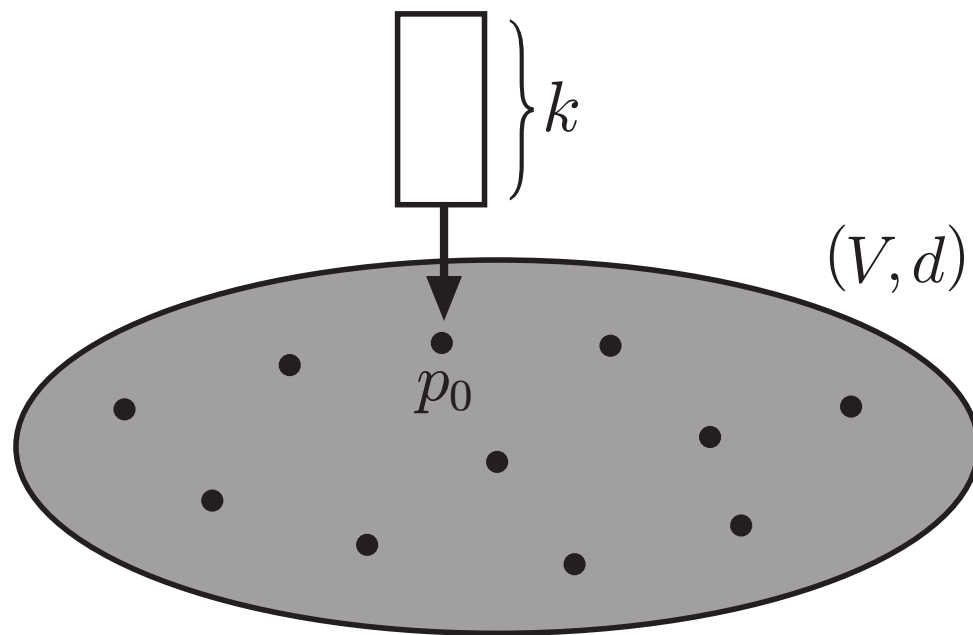
The Sorting Buffer Problem

- Input:**
- A **metric space** (V, d) .
 - A **server**, initially positioned at $p_0 \in V$.



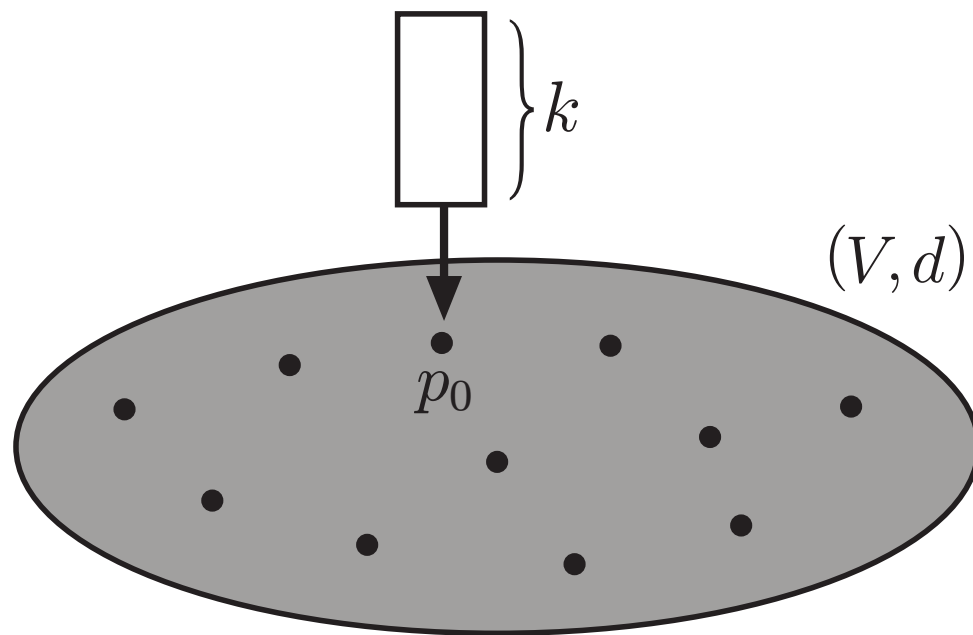
The Sorting Buffer Problem

- Input:**
- A **metric space** (V, d) .
 - A **server**, initially positioned at $p_0 \in V$.
 - A **buffer**, capable of holding k requests.



The Sorting Buffer Problem

- Input:**
- A **metric space** (V, d) .
 - A **server**, initially positioned at $p_0 \in V$.
 - A **buffer**, capable of holding k requests.
 - An **online sequence** $\sigma = \langle \sigma_1, \dots, \sigma_N \rangle$ of requests, each of which corresponds to a point in V .



The Sorting Buffer Problem

- Input:**
- A **metric space** (V, d) .
 - A **server**, initially positioned at $p_0 \in V$.
 - A **buffer**, capable of holding k requests.
 - An **online sequence** $\sigma = \langle \sigma_1, \dots, \sigma_N \rangle$ of requests, each of which corresponds to a point in V .
-
- Arriving requests **must be stored** in the buffer.
 - A pending request σ_i is served by ...
 1. **Drawing** it out of the buffer.
 2. **Moving the server** to σ_i .

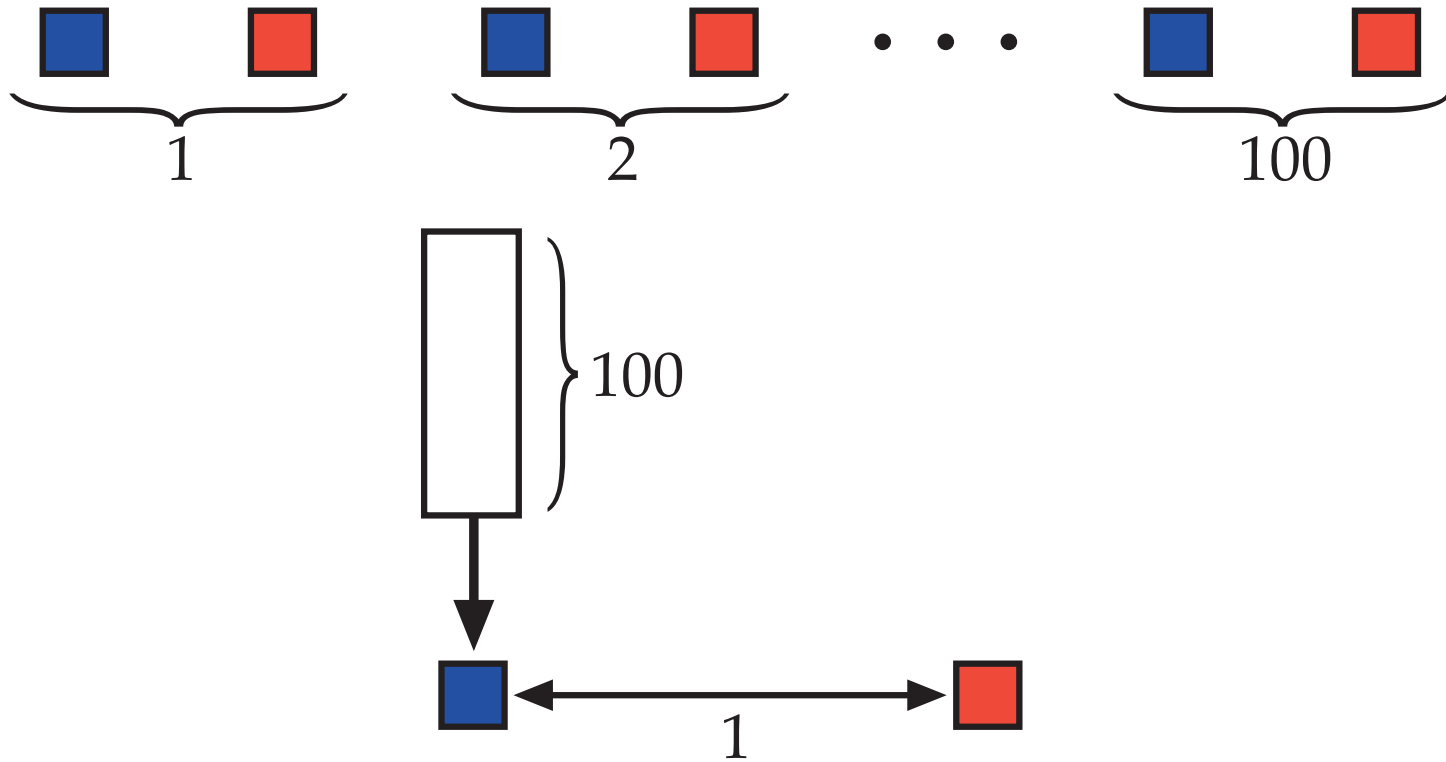
The Sorting Buffer Problem

- Input:**
- A **metric space** (V, d) .
 - A **server**, initially positioned at $p_0 \in V$.
 - A **buffer**, capable of holding k requests.
 - An **online sequence** $\sigma = \langle \sigma_1, \dots, \sigma_N \rangle$ of requests, each of which corresponds to a point in V .

Objective: Serve all input requests in a way that **minimizes the total distance** traveled by the server.

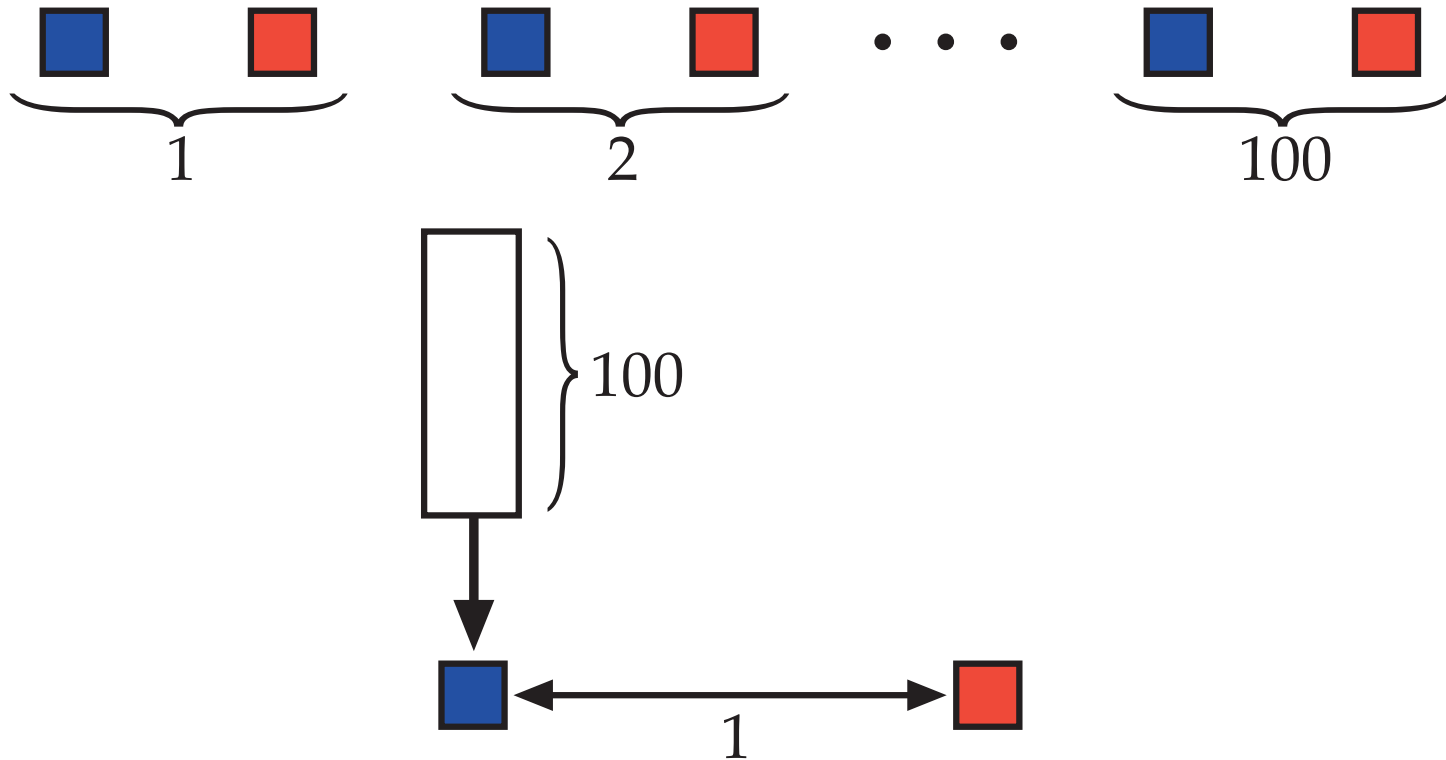
Very Simple Example: 2-Point Metric

Online sequence of alternating requests:



Very Simple Example: 2-Point Metric

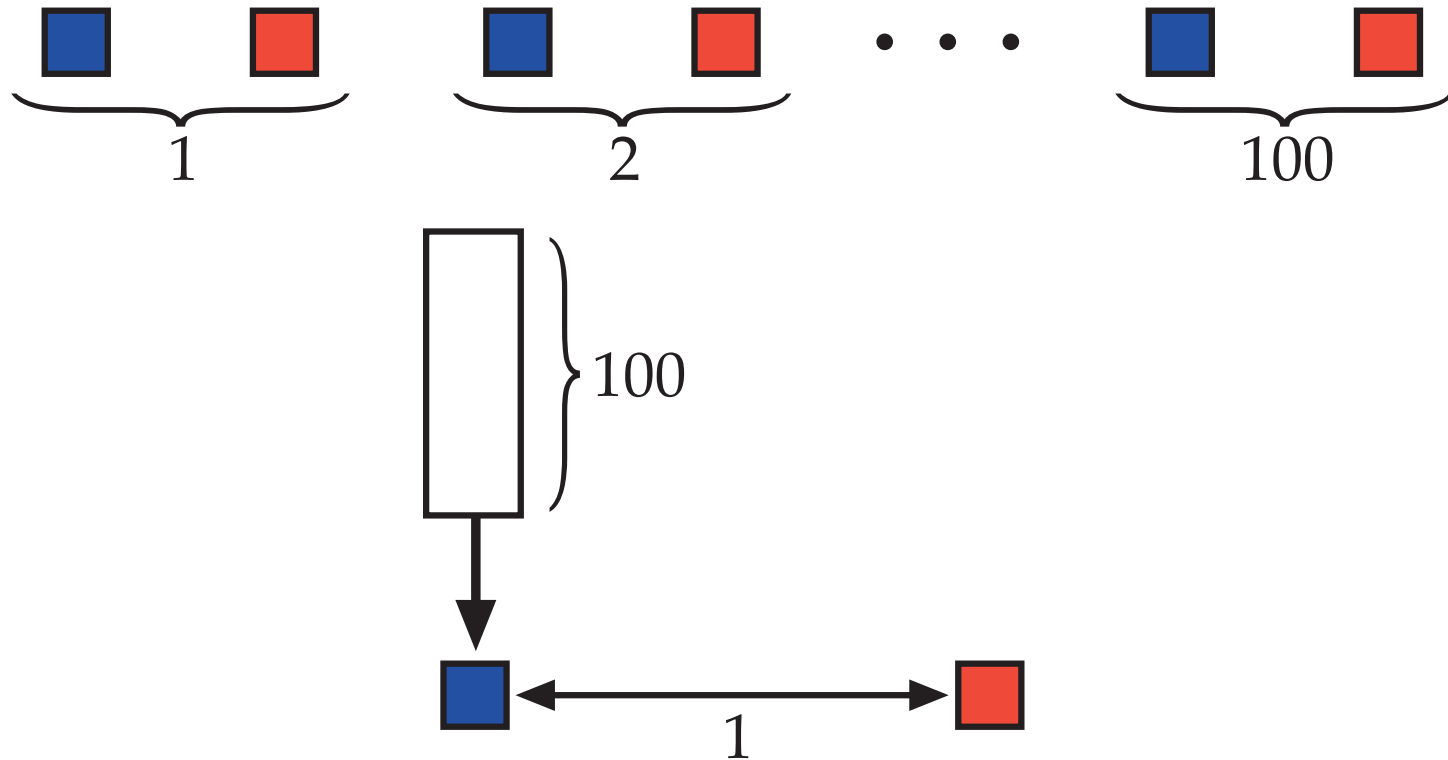
Online sequence of alternating requests:



FIFO scheduling: Move server according to original order.

Very Simple Example: 2-Point Metric

Online sequence of alternating requests:

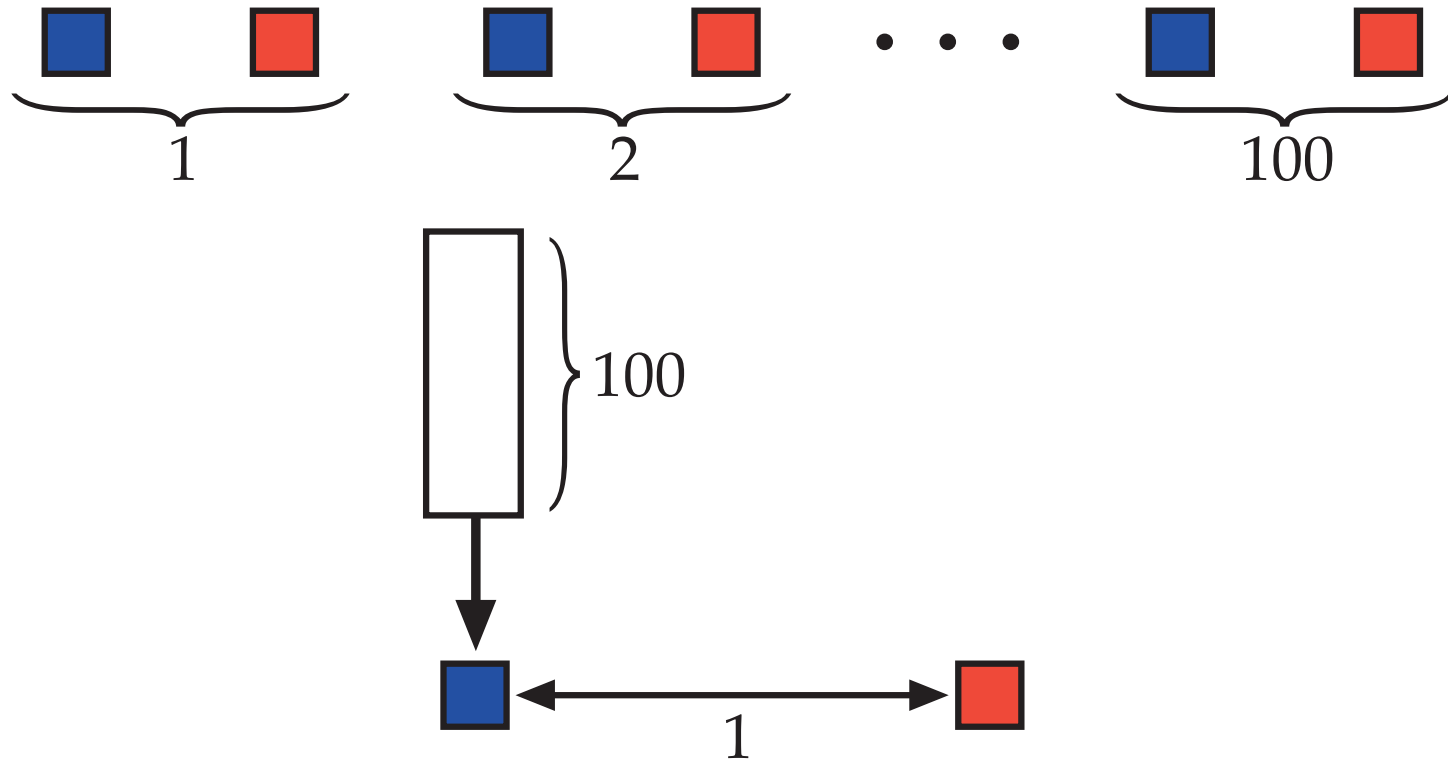


FIFO scheduling: Move server according to original order.

Total distance: 199

Very Simple Example: 2-Point Metric

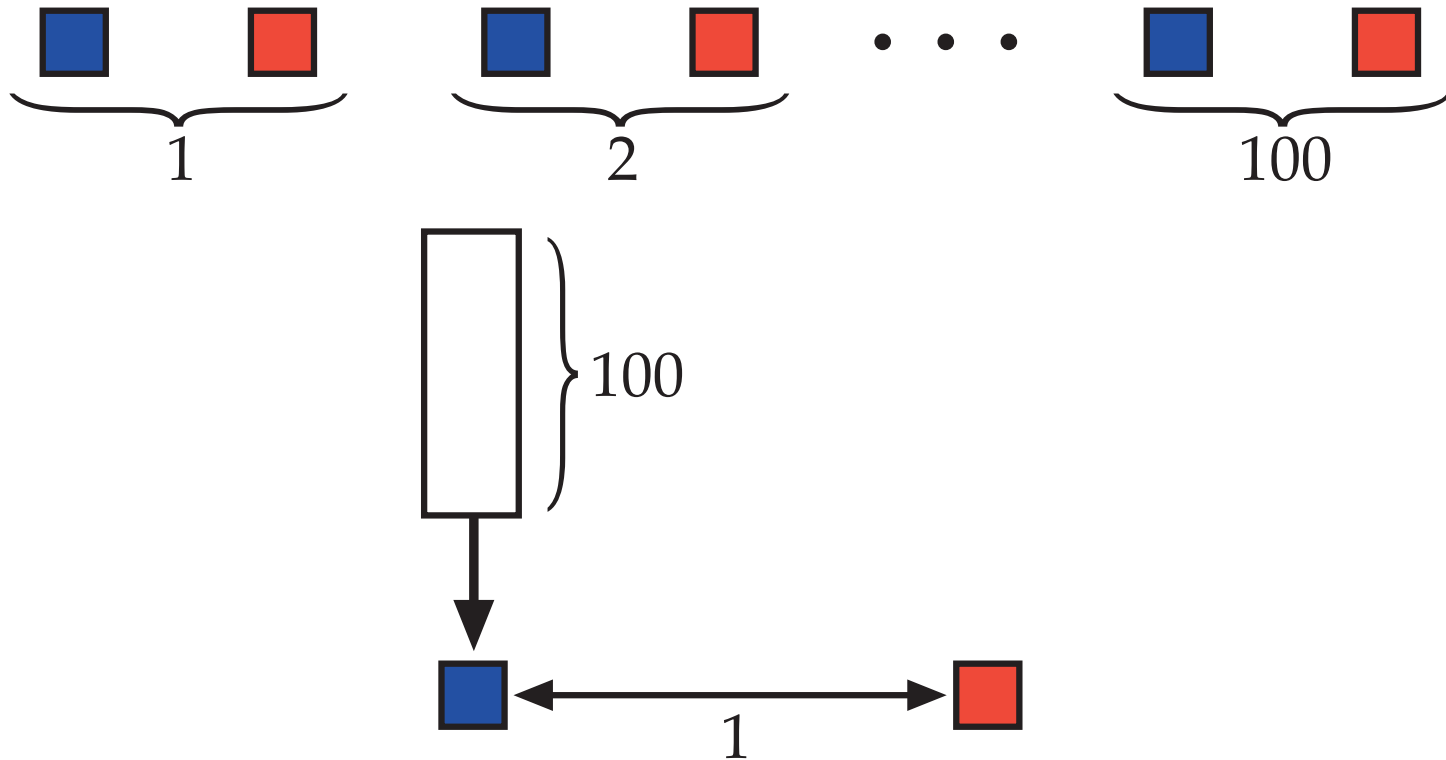
Online sequence of alternating requests:



Optimal solution: Insert ■'s to buffer while serving all ■'s; then move to ■.

Very Simple Example: 2-Point Metric

Online sequence of alternating requests:



Optimal solution: Insert ■'s to buffer while serving all ■'s; then move to ■.

Total distance: 1

Highlights of Previous Work

General metric:

- **No non-trivial results**, even for the offline version.

Highlights of Previous Work

General metric:

- **No non-trivial results**, even for the offline version.

Uniform metric:

- Räcke, Sohler & Westermann [ESA '02]:
 - Competitive ratio of $O(\log^2 k)$.
 - Lower bounds for several well-known heuristics.
- Englert & Westermann [ICALP '05]:
 - Improved competitive ratio to $O(\log k)$.
 - Algorithm works for star-like metrics.

Highlights of Previous Work

General metric:

- **No non-trivial results**, even for the offline version.

Uniform metric:

- Räcke, Sohler & Westermann [ESA '02]:
 - Competitive ratio of $O(\log^2 k)$.
 - Lower bounds for several well-known heuristics.
- Englert & Westermann [ICALP '05]:
 - Improved competitive ratio to $O(\log k)$.
 - Algorithm works for star-like metrics.

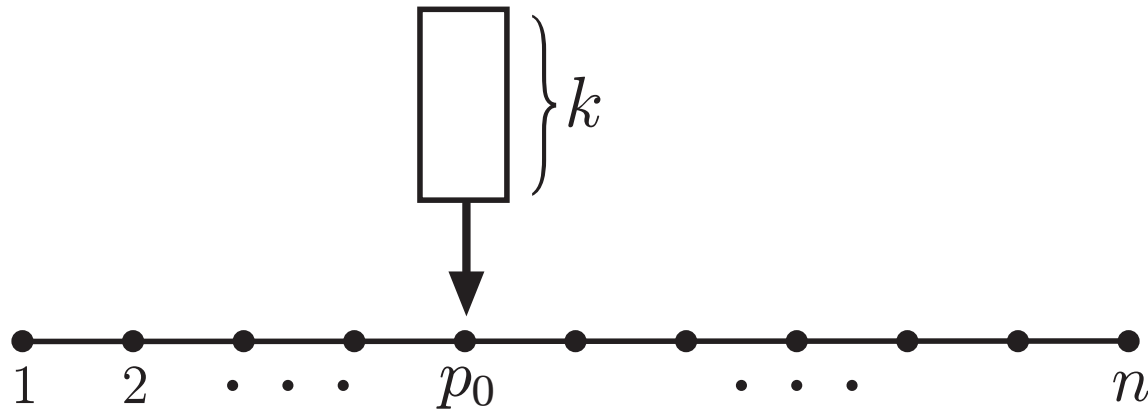
Uniform metric + maximization + offline:

- Kohrt & Pruhs [LATIN '04]: **20-approximation**.
- Bar-Yehuda & Laserson [WAOA '05]: **9-approximation**.

Evenly-Spaced Line Metrics

The special case in which ...

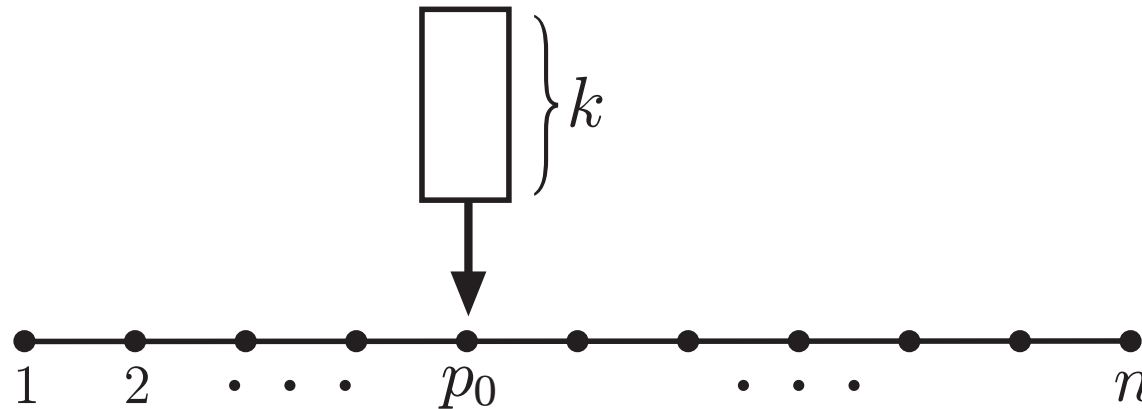
$$V = \{1, \dots, n\} \text{ and } d(p, q) = |p - q|.$$



Evenly-Spaced Line Metrics

The special case in which ...

$$V = \{1, \dots, n\} \text{ and } d(p, q) = |p - q|.$$



This setting captures the **disk arm scheduling** problem:

- line \Leftrightarrow hard disk
- points \Leftrightarrow cylinders
- server \Leftrightarrow disk arm
- request \Leftrightarrow \langle information, # cylinder \rangle block

Best Known Strategy

Khandekar and Pandit [STACS '06]:

A **randomized** online algorithm that guarantees an expected competitive ratio of $O(\log^2 n)$ against an **oblivious adversary**.

Best Known Strategy

Khandekar and Pandit [STACS '06]:

A **randomized** online algorithm that guarantees an expected competitive ratio of $O(\log^2 n)$ against an **oblivious adversary**.

- Based on **probabilistically embedding** into a distribution over binary hierarchically well-separated trees.
- Seems artificial, but **considerably simplifies** the analysis.

Best Known Strategy

Khandekar and Pandit [STACS '06]:

A **randomized** online algorithm that guarantees an expected competitive ratio of $O(\log^2 n)$ against an **oblivious adversary**.

- Based on **probabilistically embedding** into a distribution over binary hierarchically well-separated trees.
- Seems artificial, but **considerably simplifies** the analysis.

Conjecture: A deterministic strategy is unlikely to obtain a non-trivial competitive ratio.

Best Known Strategy

Khandekar and Pandit [STACS '06]:

A **randomized** online algorithm that guarantees an expected competitive ratio of $O(\log^2 n)$ against an **oblivious adversary**.

- Based on **probabilistically embedding** into a distribution over binary hierarchically well-separated trees.
- Seems artificial, but **considerably simplifies** the analysis.

Conjecture: A deterministic strategy is unlikely to obtain a non-trivial competitive ratio.

FALSE

New Upper Bound

We devise a **deterministic** online algorithm, whose competitive ratio is $O(\log n)$.

New Upper Bound

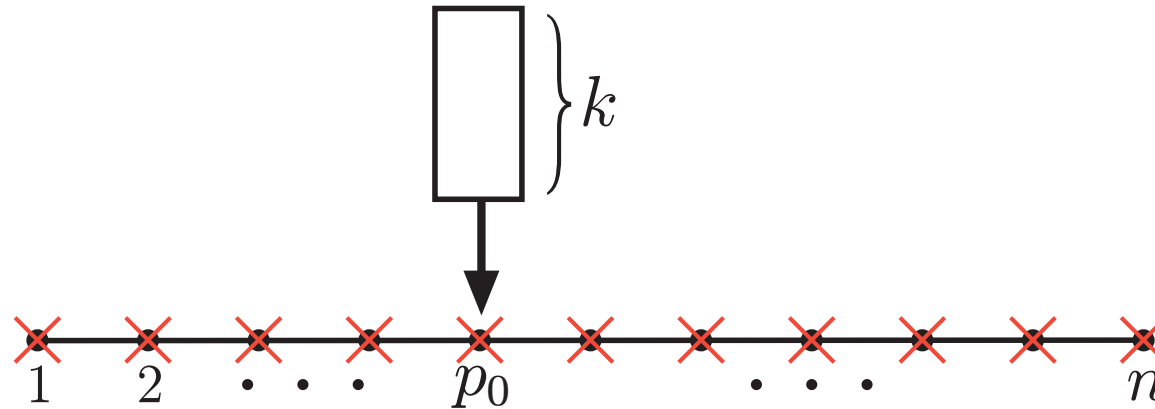
We devise a **deterministic** online algorithm, whose competitive ratio is $O(\log n)$.

Techniques:

- HSTs are replaced by **doubling partitions**.
- Interesting **charging argument**.

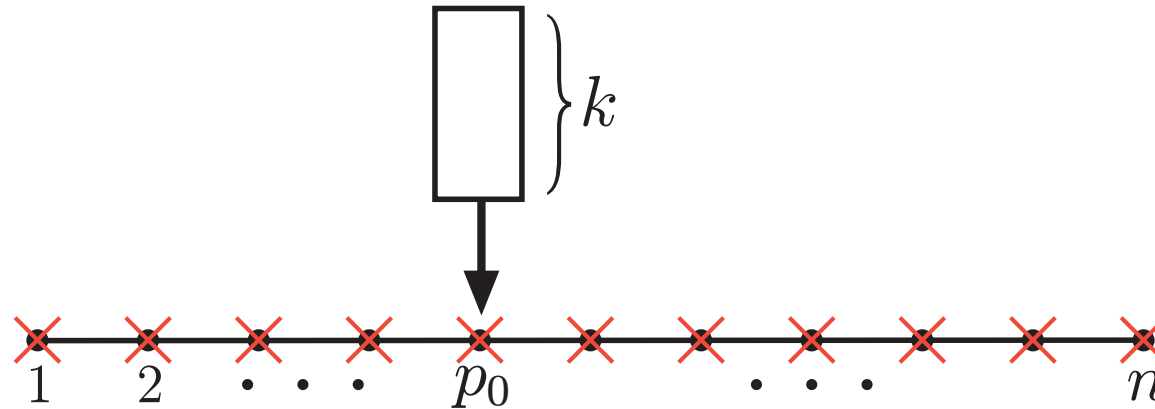
Continuous Line Metrics

Instead of $V = \{1, \dots, n\}$, we now have $V = \mathbb{R}$.



Continuous Line Metrics

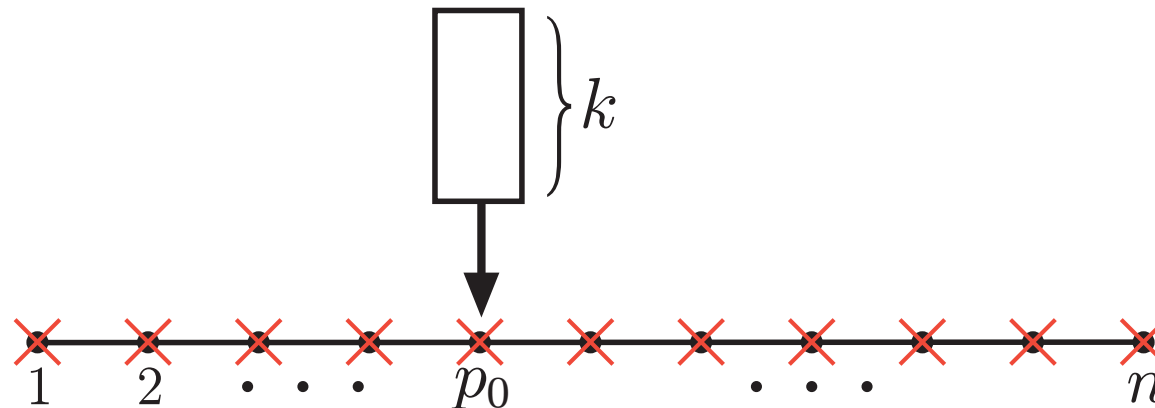
Instead of $V = \{1, \dots, n\}$, we now have $V = \mathbb{R}$.



We achieve a competitive ratio of $O(\log N \log \log N)$.

Continuous Line Metrics

Instead of $V = \{1, \dots, n\}$, we now have $V = \mathbb{R}$.



We achieve a competitive ratio of $O(\log N \log \log N)$.

Techniques:

- Online guessing of diameter and sequence length.
- Interesting discretization and gluing methods.

A Deterministic Lower Bound

Khandekar and Pandit [STACS '06]:

Posed the task of attaining **lower bounds** on the achievable **competitive ratio** as a foundational research objective.

A Deterministic Lower Bound

Khandekar and Pandit [STACS '06]:

Posed the task of attaining **lower bounds** on the achievable **competitive ratio** as a foundational research objective.

We prove that the competitive ratio of any **deterministic** online algorithm is at least $\frac{2+\sqrt{3}}{\sqrt{3}} \approx 2.15$.

Concluding Remarks

- Our algorithms can be easily modified to attain a competitive ratio of $O(\log \Delta)$, where Δ denotes the **aspect ratio** of the given line metric.

Concluding Remarks

- Our algorithms can be easily modified to attain a competitive ratio of $O(\log \Delta)$, where Δ denotes the **aspect ratio** of the given line metric.
- **Close the gap** between $O(\log n)$ and 2.15, or try to obtain a **performance guarantee of $o(k)$** .

Concluding Remarks

- Our algorithms can be easily modified to attain a competitive ratio of $O(\log \Delta)$, where Δ denotes the **aspect ratio** of the given line metric.
- **Close the gap** between $O(\log n)$ and 2.15, or try to obtain a **performance guarantee of $o(k)$** .
- Any non-trivial result for **general metrics**.

Concluding Remarks

- Our algorithms can be easily modified to attain a competitive ratio of $O(\log \Delta)$, where Δ denotes the **aspect ratio** of the given line metric.
- **Close the gap** between $O(\log n)$ and 2.15, or try to obtain a **performance guarantee of $o(k)$** .
- Any non-trivial result for **general metrics**.

Thank You