

Lowerbounds for the Online Minimum Matching Problem on the Line

Maximillian C.W. Bender

Advisor: Christine Chung

Connecticut College Department of Computer Science

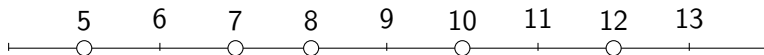


CONNECTICUT
COLLEGE

An Example

| Ski Size | Renter's Size | Difference |
|----------|---------------|------------|
| 5 | | |
| 7 | | |
| 8 | | |
| 10 | | |
| 12 | | |

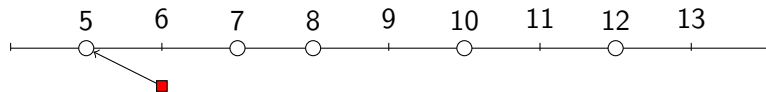
Renter's Size Sequence:



An Example

| Ski Size | Renter's Size | Difference |
|----------|---------------|------------|
| 5 | 6 | 1 |
| 7 | | |
| 8 | | |
| 10 | | |
| 12 | | |
| | | Sum: 1 |
| | | Max: 1 |

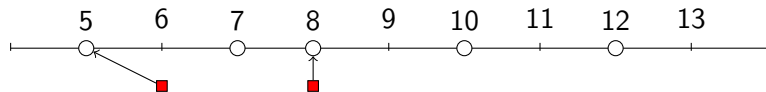
Renter's Size Sequence: (6,



An Example

| Ski Size | Renter's Size | Difference |
|----------|---------------|------------|
| 5 | 6 | 1 |
| 7 | | |
| 8 | 8 | 0 |
| 10 | | |
| 12 | | |
| | | Sum: 1 |
| | | Max: 1 |

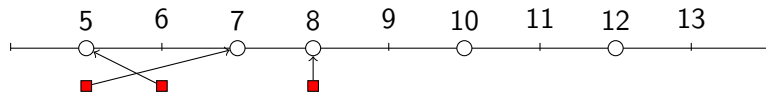
Renter's Size Sequence: (6, 8,



An Example

| Ski Size | Renter's Size | Difference |
|----------|---------------|------------|
| 5 | 6 | 1 |
| 7 | 5 | 2 |
| 8 | 8 | 0 |
| 10 | | |
| 12 | | |
| | | Sum: 3 |
| | | Max: 2 |

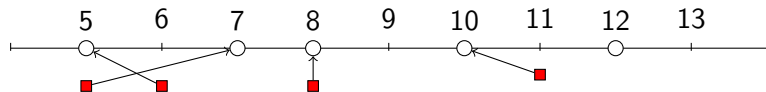
Renter's Size Sequence: (6, 8, 5,



An Example

| Ski Size | Renter's Size | Difference |
|----------|---------------|------------|
| 5 | 6 | 1 |
| 7 | 5 | 2 |
| 8 | 8 | 0 |
| 10 | 11 | 1 |
| 12 | | |
| | | Sum: 4 |
| | | Max: 2 |

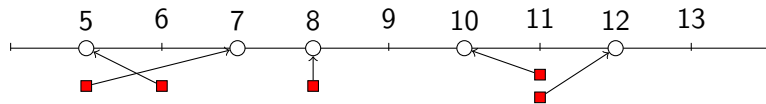
Renter's Size Sequence: (6, 8, 5, 11,



An Example

| Ski Size | Renter's Size | Difference |
|----------|---------------|------------|
| 5 | 6 | 1 |
| 7 | 5 | 2 |
| 8 | 8 | 0 |
| 10 | 11 | 1 |
| 12 | 11 | 1 |
| | | Sum: 5 |
| | | Max: 2 |

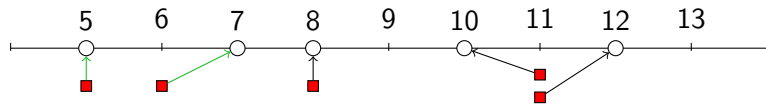
Renter's Size Sequence: (6, 8, 5, 11, 11)



An Example

| Ski Size | Renter's Size | Difference |
|----------|---------------|------------|
| 5 | 5 | 0 |
| 7 | 6 | 1 |
| 8 | 8 | 0 |
| 10 | 11 | 1 |
| 12 | 11 | 1 |
| | | Sum: 3 |
| | | Max: 1 |

Renter's Size Sequence: (6, 8, 5, 11, 11)



Questions for the Online Matching Model

- 1 What is the best general approach to minimizing the total or maximal difference between the sizes? Can we establish an **upperbound** to the approach's cost?
- 2 What is the best we could hope to achieve with any algorithm to minimize the total or maximal difference? Can we fix a **lowerbound** for any algorithm's performance?

Formalism

Problem Definition

- Given:
 - A set $S \subset \mathbb{R}$ of n servers
 - A sequence of requests $R = (r_1, r_2, \dots, r_n) \subset \mathbb{R}$ whose locations are unknown prior to arrival.

- Given:
 - A set $S \subset \mathbb{R}$ of n servers
 - A sequence of requests $R = (r_1, r_2, \dots, r_n) \subset \mathbb{R}$ whose locations are unknown prior to arrival.
- Goal: To construct a bijective function $f : R \rightarrow S$ that best approximates either of the following objective functions:
 - Min-Weight Objective: $f = \operatorname{argmin}_{g:R \rightarrow S} \left(\sum_{i=1}^n |g(r_i) - r_i| \right)$
 - Bottleneck Objective: $f = \operatorname{argmin}_{g:R \rightarrow S} \left(\max_{i \in \{1, \dots, n\}} |g(r_i) - r_i| \right)$

Definition (Cost)

We denote the *matching function* of any algorithm A on a specific instance I by $A(I) = f$ and the *cost* of any matching function as

$$c(f) = \sum_{i=1}^n |f(r_i) - r_i|$$

for the min weight objective and for the bottleneck objective

$$c(f) = \max_{i \in \{1, \dots, n\}} |f(r_i) - r_i|.$$

Definition (Deterministic vs. Randomized)

We denote the set of deterministic algorithms for this problem \mathcal{A}_D . We denote the set of randomized algorithms \mathcal{A} , where $\mathcal{A}_D \subset \mathcal{A}$.

Definition (Competitive Ratio for Deterministic Algorithms)

We define the *competitive ratio* ρ of any *deterministic* algorithm $A \in \mathcal{A}_D$ by

$$\rho = \max_{I \in \mathcal{I}} \left(\frac{c(A(I))}{\text{OPT}(I)} \right)$$

where \mathcal{I} denotes the set of all possible instances for the problem.

Definition (Competitive Ratio for Randomized Algorithms)

We define the *competitive ratio* ρ of any *randomized* algorithm $A \in \mathcal{A}$ by

$$\rho = \max_{I \in \mathcal{I}} \left(\mathbb{E} \left[\frac{c(A(I))}{\text{OPT}(I)} \right] \right)$$

where \mathcal{I} denotes the set of all possible instances for the problem.

Theorem (Yao's Principle)

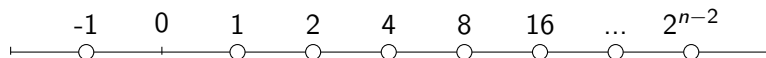
Given a probability distribution p over $\mathcal{A}_{\mathcal{D}}$ and q over \mathcal{I} , where A is some algorithm chosen according to p and I some instance chosen according to q , then

$$\max_{i \in \mathcal{I}} \mathbb{E}[c(A(i))] \geq \min_{a \in \mathcal{A}_{\mathcal{D}}} \mathbb{E}[c(a(I))].$$

Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

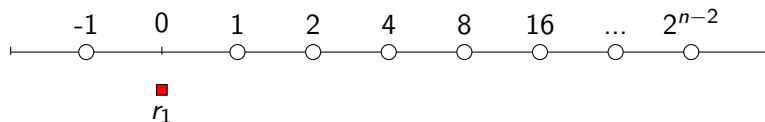
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

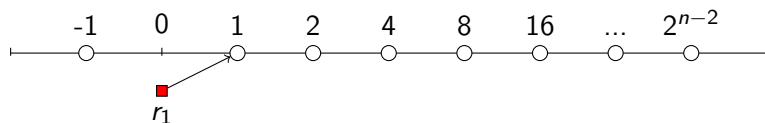
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

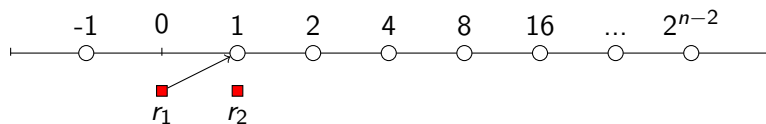
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

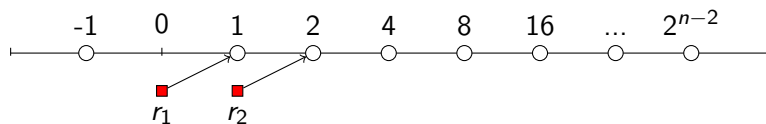
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

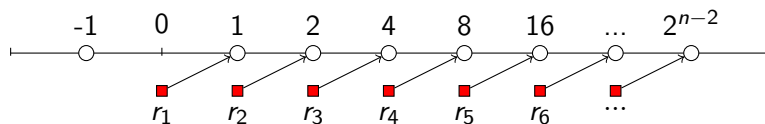
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

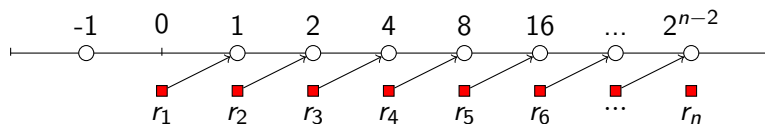
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

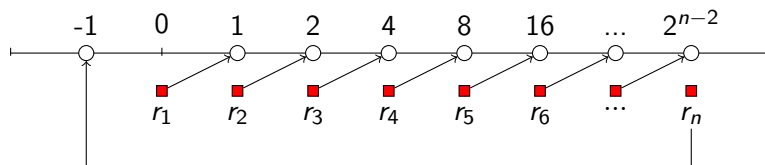
Consider the following instance I with n servers:



Lowerbound of GREEDY

We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

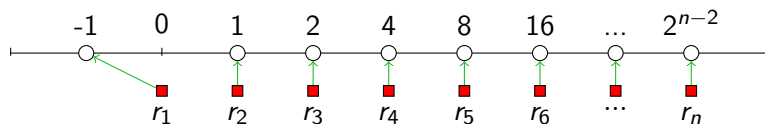
Consider the following instance I with n servers:



Lowerbound of GREEDY

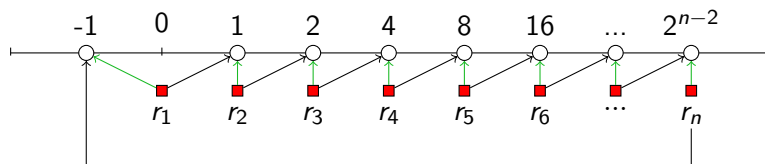
We will define the deterministic GREEDY algorithm as the algorithm which sends each request to its closest available server at the time of arrival.

Consider the following instance I with n servers:



Lowerbound of GREEDY

Min-Weight



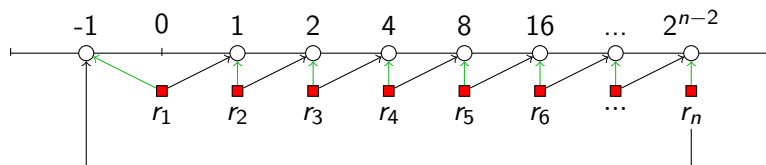
$$\begin{aligned}c(\text{GREEDY}(I)) &= 1 + \sum_{i=1}^{n-2} (2^i - 2^{i-1}) + 2^{n-2} + 1 \\ &= 1 + 2 \times 2^{n-2} = 1 + 2^{n-1}\end{aligned}$$

$$\text{OPT}(I) = 1$$

$$\rho \geq 2^{n-1}$$

Lowerbound of GREEDY

Bottleneck



$$c(\text{GREEDY}(I)) = 2^{n-2} + 1$$

$$\text{OPT}(I) = 1$$

$$\rho \geq 2^{n-2}$$

Previous Results

Deterministic:

| | Lowerbound | Best known Algorithm | Competitive Ratio |
|-------------|--------------------|----------------------|-------------------|
| Total: | $9 + \epsilon$ [1] | PERMUTATION [4][5] | $2n - 1$ |
| Bottleneck: | $1.5n$ [3] | PERMUTATION | $2n - 1$ |

Randomized:

| | Lowerbound | Best known Algorithm | Competitive Ratio |
|-------------|------------|----------------------|-------------------|
| Total: | — | HARMONIC [2] | $O(\log(n))$ |
| Bottleneck: | — | PERMUTATION | $2n - 1$ |

Deterministic:

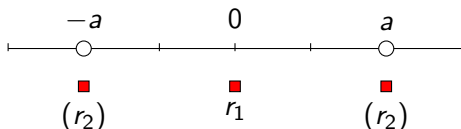
| | Lowerbound | Best known Algorithm | Competitive Ratio |
|-------------|--------------------|----------------------|-------------------|
| Total: | $9 + \epsilon$ [1] | PERMUTATION [4][5] | $2n - 1$ |
| Bottleneck: | $1.5n$ [3] | PERMUTATION | $2n - 1$ |

Randomized:

| | Lowerbound | Best known Algorithm | Competitive Ratio |
|-------------|---------------|----------------------|-------------------|
| Total: | 2 | HARMONIC [2] | $O(\log(n))$ |
| Bottleneck: | $\frac{n}{2}$ | PERMUTATION | $2n - 1$ |

Lowerbound for Min-Weight

Consider the following instance with $n = 2$ servers:



The request sequence is $R = (0, \pm a)$, ie the second request is either $\{-a, a\}$ with uniform probability.

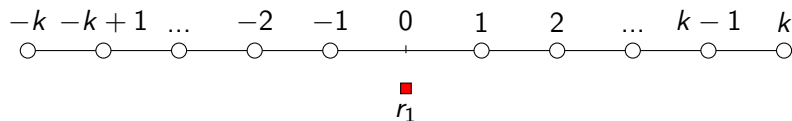
Hence for any algorithm $A \in \mathcal{A}$:

$$\mathbb{E}[c(A(I))] = \frac{1}{2}(a + 0) + \frac{1}{2}(a + 2a) = \frac{4a}{2} = 2a.$$

However the optimal cost will always be just a , hence $\rho \geq \frac{2a}{a} = 2$.

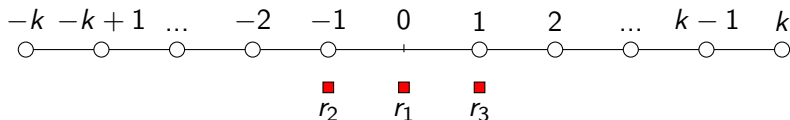
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



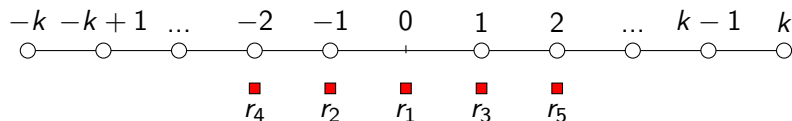
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



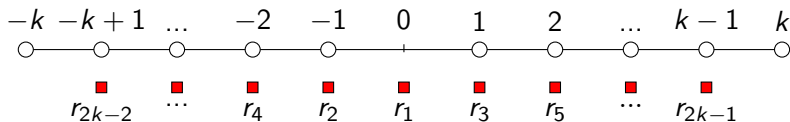
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



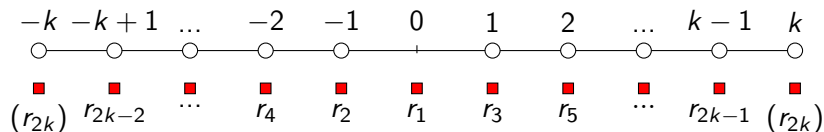
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



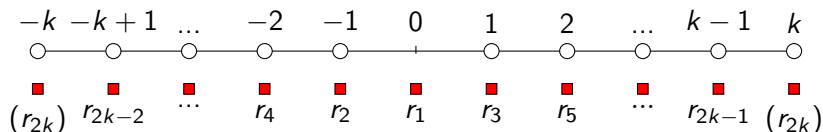
Lowerbound for Randomized Algorithms under the Bottleneck Objective

Consider the following instance with $n = 2k$ servers:



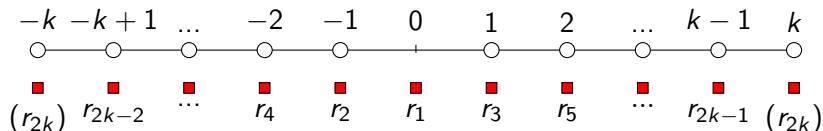
The final request r_{2k} is either $\{-k, k\}$ with uniform probability.

Lowerbound for Randomized Algorithms under the Bottleneck Objective



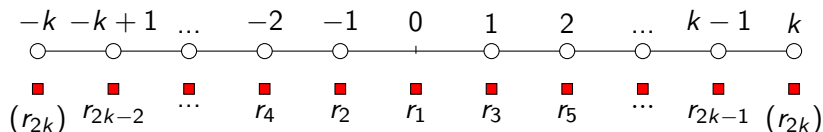
- The important thing here is actually only the last request

Lowerbound for Randomized Algorithms under the Bottleneck Objective



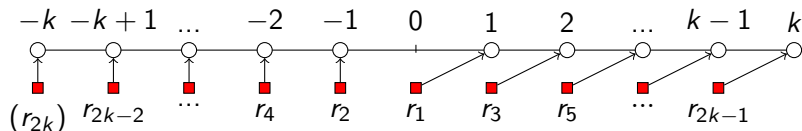
- The important thing here is actually only the last request
- After the first $2k - 1$ requests, there will only be one available server, call it s

Lowerbound for Randomized Algorithms under the Bottleneck Objective



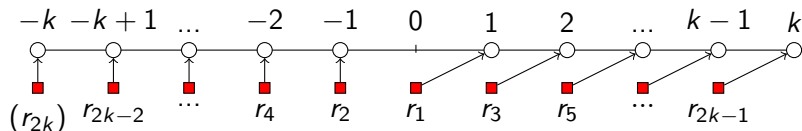
- The important thing here is actually only the last request
- After the first $2k - 1$ requests, there will only be one available server, call it s
- $E[|s - r_{2k}|] = \frac{1}{2}(k - s) + \frac{1}{2}(s + k) = k$

Lowerbound for Randomized Algorithms under the Bottleneck Objective



- The important thing here is actually only the last request
- After the first $2k - 1$ requests, there will only be one available server, call it s
- $E[|s - r_{2k}|] = \frac{1}{2}(k - s) + \frac{1}{2}(s + k) = k$
- The optimal bottleneck edge cost will always be just 1

Lowerbound for Randomized Algorithms under the Bottleneck Objective



- The important thing here is actually only the last request
- After the first $2k - 1$ requests, there will only be one available server, call it s
- $E[|s - r_{2k}|] = \frac{1}{2}(k - s) + \frac{1}{2}(s + k) = k$
- The optimal bottleneck edge cost will always be just 1
- Hence $\rho \geq k = \frac{n}{2}$

Concluding Remarks

Deterministic:

| | Lowerbound | Best known Algorithm | Competitive Ratio |
|-------------|--------------------|----------------------|-------------------|
| Total: | $9 + \epsilon$ [1] | PERMUTATION [4][5] | $2n - 1$ |
| Bottleneck: | $1.5n$ [3] | PERMUTATION | $2n - 1$ |

Randomized:

| | Lowerbound | Best known Algorithm | Competitive Ratio |
|-------------|---------------|----------------------|-------------------|
| Total: | 2 | HARMONIC [2] | $O(\log(n))$ |
| Bottleneck: | $\frac{n}{2}$ | PERMUTATION | $2n - 1$ |

Concluding Remarks

Theorem (No Skip Necessary)

Given any instance of the problem for the minimum weight objective, there exists an optimal min-weight matching function $f : R \rightarrow S$ that satisfies $f(r_j) \in N_f(r_j)$, where $N_f(r_j)$ denotes the neighboring unmatched servers of r_j on the line. Formally, by letting

$$S_j = S \setminus \{f(r_1), \dots, f(r_{j-1})\},$$

$$L_j = \max \{s \mid s \in S_j, s \leq r_j\}, \text{ and}$$

$$R_j = \min \{s \mid s \in S_j, s \geq r_j\},$$

then $N_f(r_j) = \{L_j, R_j\}$.



B. Fuchs, W. Hochstättler, and W. Kern.

Online matching on a line.

In *Electronic Notes In Discrete Mathematics*. Elsevier, 2003.



Anupam Gupta and Kevin Lewi.

The online metric matching problem for doubling metrics.

In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 424–435. Springer Berlin Heidelberg, 2012.



R. Idury and A. A. Schäffer.

A better lower bound for on-line bottleneck matching.

<http://www.ncbi.nlm.nih.gov/core/assets/cbb/files/Firehouse.pdf>.

Accessed: 2015-09-7.



B. Kalyanasundaram and K. Pruhs.

Online weighted matching.

J. Algorithms, 14(3):478–488, 1993.



S. Khuller, S. G. Mitchell, and V. V. Vazirani.

On-line algorithms for weighted bipartite matching and stable marriages.

Theoretical Computer Science, 127:251–264, 1994.