The Power of Greedy for Online Minimum Cost Matching on the Line

ERIC BALKANSKI, Columbia University, IEOR department, USA
YURI FAENZA, Columbia University, IEOR department, USA
NOÉMIE PÉRIVIER, Columbia University, IEOR department, USA

In the online minimum cost matching problem, there are n servers and, at each of n time steps, a request arrives and must be irrevocably matched to a server that has not yet been matched, with the goal of minimizing the sum of the distances between the matched pairs. Online minimum cost matching is a central problem in applications such as ride-hailing platforms and food delivery services. Despite achieving a worst-case competitive ratio that is exponential in n even on the line, the simple greedy algorithm, which matches each request to its nearest available server, performs well in practice and has a number of attractive features such as strategyproofness. A major question is thus to explain greedy's strong empirical performance. In this paper, we aim to understand the performance of greedy on the line over instances that are at least partially random.

When both the requests and the servers are drawn uniformly and independently from $[0, 1]$, we obtain a constant competitive ratio for greedy, which improves over the previously best-known bound of $O(\sqrt{n})$ for greedy in this setting. We also show that this constant competitive ratio also holds in the excess supply setting where there is a linear excess of servers, which improves over the previously best-known bound of $O(\log^3 n)$ for greedy in this setting.

We moreover show that in the semi-random model where the requests are still drawn uniformly and independently but where the servers are chosen adversarially, greedy achieves an $O(\log n)$ competitive ratio. Even though this one-sided randomness allows a large improvement in greedy's competitive ratio compared to the model where the requests are fully adversarial or arrive in a random order, we show that it is not sufficient to obtain a constant competitive ratio by giving a tight $\Omega(\log n)$ lower bound. These results invite further investigation about how much randomness is necessary and sufficient to obtain strong theoretical guarantees for the greedy algorithm for online minimum cost matching, on the line and beyond. A full version of this paper can be found at https://arxiv.org/abs/2210.03166.

CCS Concepts: • **Theory of computation → Online algorithms**; • **Computing methodologies → Model development and analysis**; • **Mathematics of computing → Graph algorithms**.

Additional Key Words and Phrases: online algorithm, online matching, beyond-worst case analysis, greedy algorithm, semi-random model

ACM Reference Format:
Eric Balkanski, Yuri Faenza, and Noémie Périvier. 2023. The Power of Greedy for Online Minimum Cost Matching on the Line. In *Proceedings of the 24th ACM Conference on Economics and Computation (EC '23), July 9–12, 2023, London, United Kingdom.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3580507.3597794

1 Introduction

Matching problems are a core area of discrete optimization. In the 90s, a seminal paper by Karp et al. [1990] introduced online bipartite maximum matching problems and showed that, in the worst-case scenario, no deterministic algorithm can beat a simple greedy procedure, and no randomized

EC '23, July 9–12, 2023, London, United Kingdom

ACM ISBN 979-8-4007-0104-7/23/07…$15.00
https://doi.org/10.1145/3580507.3597794

algorithm can beat *ranking*, which is a greedy procedure preceded by a random shuffling of the order of the nodes. These elegant results and their natural application to online advertising spurred much research, especially from the late 2000s on (see, e.g., [Mehta et al., 2013] and the references therein for a survey). While more complex algorithms have been devised for models other than worst-case analysis, greedy techniques are often used as a competitive benchmark for comparisons, see, e.g., [Feldman et al., 2010, Li et al., 2020, Xu et al., 2019].

In the last few years, motivated by the surge of ride-sharing platforms, a second online matching paradigm has received much attention: online (bipartite) minimum cost matching. In this class of problems, one side of the market is composed of *servers* (sometimes called *drivers*) and is fully known at time 0. Nodes from the other side, often called *requests* or *customers*, arrive one at a time. When request i arrives, we must match it to one of the servers j, and incur a cost c_{ij}. Server j is then removed from the list of available servers, and the procedure continues. The goal is to minimize the total cost of the matching.

Given the motivating application to ride-sharing, it is natural to impose the condition that both servers and requests belong to some metric space (e.g., [Gairing and Klimm, 2019, Kalyanasundaram and Pruhs, 1993, Kanoria, 2022, Raghvendra, 2016, Tsai et al., 1994]). Many algorithms in this area involve non-trivial, in some cases computationally expensive, procedures like randomized tree embeddings [Bansal et al., 2007, Meyerson et al., 2006], iterative segmentation of the space [Kanoria, 2022] or primal-dual arguments based on the computation of offline optimal matchings at each time step [Raghvendra, 2018]. Other algorithms use randomization to bypass worst-case scenarios for deterministic algorithms [Gupta and Lewi, 2012].

The predominant objective of this line work has been to design algorithms that achieve the strongest possible performance guarantees in terms of quality of the solution found, which is measured by an algorithm's competitive ratio. However, there are other important considerations when deploying systems that match individuals in real time, such as simplicity, strategyproofness, running time, and explainability. An extremely simple algorithm that is highly desirable with respect to all these factors is the greedy algorithm, also called *nearest neighbor*, that matches each incoming request to the closest available server. But does it perform well?

Somehow surprisingly, this algorithm often works very well in practice: experiments have shown that greedy was more effective than other existing algorithms in most tests and has outstanding scalability [Tong et al., 2016]. This performance substantiates the choice of many ride-sharing platforms to actually implement greedy procedures, in combination with other techniques [Brown, 2016, Jackson, 2019]. However, current theory exhibits a mismatch with such strong computational results: if we assume that n servers and n requests are adversarially placed on a line, the greedy algorithm only achieves a $2^n - 1$ competitive ratio [Kalyanasundaram and Pruhs, 1993, Tsai et al., 1994]. It is therefore important to develop a theory that closes the gap with practice and gives solid ground to the use of the greedy algorithm. This motivates the first guiding question of the paper.

> *Can we find a theoretical justification for the strong practical performance of the greedy algorithm for online minimum cost matching problems?*

A standard approach to the question above is to make a distributional assumption on the input. Obviously, stronger assumptions may lead to stronger positive results – but such assumptions may not be verified in practice. Ideally, we would like to identify the hypotheses that are necessary to guarantee a strong performance for the greedy procedure. These results can provide important guidance to practitioners: depending on whether or not they believe such hypothesis to be verified by their data, they can choose to either apply the greedy algorithm or to resort to a more refined procedure. This discussion motivates the second guiding question of the paper.

What are necessary and sufficient assumptions to guarantee that the greedy procedure outputs a solution whose quality is asymptotically optimal?

These questions have important implications since they aim to characterize the scenarios where a simple greedy algorithm can be used instead of significantly more complex algorithms for a problem central to multiple large modern markets such as ride-sharing and food delivery. Understanding the strong practical performance of simple algorithms has motivated a lot of work on beyond the worst-case analysis of algorithms. Some examples include using properties such as curvature, stability, sharpness, and smoothness to obtain improved guarantees for greedy for submodular maximization [Chatziafratis et al., 2017, Conforti and Cornuéjols, 1984, Pokutta et al., 2020, Rubinstein and Zhao, 2022] and different semi-random models for analyzing k-means for clustering [Arthur et al., 2009, Manthey and Röglin, 2013], local search for the traveling salesman problem [Balkanski et al., 2022, Englert et al., 2014, 2016, Künnemann and Manthey, 2015], and greedy for online *maximum* matching [Arnosti, 2022, Devanur et al., 2011, Goel and Mehta, 2008, Mastin and Jaillet, 2013]. In the context of online *minimum* cost matching, our understanding of the performance of greedy is very limited. Despite its simplicity, greedy is hard to analyze because a greedy match at some time step can have complex consequences on the available servers in a different region at a much later time step. In other words, "the state of the system under the standard greedy algorithm is hard to keep track of analytically" [Kanoria, 2021].

As a step towards understanding the power and limits of greedy, we focus on a fundamental, deceptively simple setting, which is in fact one of the most studied in the area: online minimum cost matching on the line. Despite much work [Akbarpour et al., 2022, Fuchs et al., 2003, Gupta and Lewi, 2012, Gupta et al., 2020, Koutsoupias and Nanavati, 2004, Megow and Nölke, 2020, Nayyar and Raghvendra, 2017, Peserico and Scquizzato, 2021, Raghvendra, 2018], the performance of simple algorithms for this model are far from being understood.

We first consider the *fully random* model, where the n servers and n requests are all drawn uniformly and independently from $[0, 1]$. In this model, the best known bound on the competitive ratio of greedy is a trivial $O(\sqrt{n})$ bound,[1] and there are more sophisticated algorithms such as hierarchical greedy [Kanoria, 2022] and fair-bias [Gupta et al., 2019] that are constant competitive in Euclidean spaces and on the line, respectively.[2] Our first main result settles the asymptotic performance of greedy for matching on the line in the fully random model by showing that greedy achieves a constant competitive ratio.

Theorem 1.1. *For online matching on the line in the fully random model, the greedy algorithm achieves a constant competitive ratio.*

A main benefit of greedy is that it is customer-strategyproof, meaning that the customers arriving online have no incentive to misreport the location of their requests. We note that this result improves the best-known competitive ratio of any mechanism that is customer-strategyproof from $O(\sqrt{n})$ to constant for this setting (in fact, we are not aware of any non-trivial customer-strategyproof mechanism besides greedy). We refer to the full version of the paper for a discussion and a formal definition of customer-strategyproofness.

We show that this constant competitiveness of greedy also holds in the *fully random $\epsilon-$excess model*, for every constant $\epsilon > 0$. This is a modification of the fully random model where there is a linear excess of servers, i.e., $(1 + \epsilon)n$ servers. This results improves over the previously best-known competitive ratio for greedy of $O(\log^3 n)$ in this setting [Akbarpour et al., 2022].

[1]There is a known $\Omega(\sqrt{n})$ bound on the optimal cost (see, e.g., [Tsai et al., 1994]) and the cost of any algorithm is trivially upper bounded by n.

[2]Note that hierarchical greedy is constant competitive only for $d = 1$ or $d \geq 3$. For $d = 2$, Kanoria [2022] shows that an adapted version of the gravitational matching algorithm by Holden et al. [2021] is constant competitive.

Theorem 1.2. *For any constant $\epsilon > 0$, greedy is $O(1)$-competitive in the fully random ϵ-excess model.*

It is widely acknowledged (see, e.g., [Feige, 2021]) that i.i.d. instances often do not resemble "real" instances. We next therefore consider whether strong guarantees for greedy can also be obtained in a semi-random model. In particular, we consider a model that we call the *random requests model* where the n servers are adversarially chosen and the requests are, as in the fully random model, drawn uniformly and independently. In ride-sharing, this is motivated by the fact that there have been examples of drivers that behave adversarially to increase the prices of the rides, see, e.g., [Hamilton, 2019]. Our next result shows that greedy is logarithmic competitive in the random requests model.

Theorem 1.3. *For online matching on the line in the random requests model, the greedy algorithm achieves an $O(\log n)$-competitive ratio.*

In the model where the servers and requests are chosen adversarially but where the arrival order is random, $O(n)$ and $\Omega(n^{0.22})$ upper and lower bounds are known for the competitive ratio of greedy [Gairing and Klimm, 2019]. Combined with this $\Omega(n^{0.22})$ lower bound, our result shows that the performance of greedy improves exponentially when the locations of the requests are also random. We note that hierarchical greedy only achieves a polynomial competitive ratio in the random requests model (see the full version of the paper). Our last main result shows that this competitive ratio of greedy in the random requests model is tight.

Theorem 1.4. *For online matching on the line in the random requests model, the greedy algorithm achieves an $\Omega(\log n)$-competitive ratio.*

Combined with Theorem 1.3, we obtain that greedy is $\Theta(\log n)$-competitive in the random requests model. The combination of our four results give a first partial characterization of the scenarios in which greedy is guaranteed to perform well for online minimum cost matching, but there remain many intriguing questions. In particular, we believe that it would be interesting to study semi-random requests and/or semi-random servers, for example, in a model where some fraction of the servers are adversarial and some fraction are random. Another interesting model, especially in the context of ride-sharing, would be one where the location of a small number of servers can be chosen (i.e., a mix of best-case and worst-case). Considering more general metric spaces beyond the line is of course also a direction for future work. Finally, it would be interesting to explore empirically which semi-random models exhibit a structure that most closely resembles the structure of real-world instances.

1.1 Technical overview

The main difficulty in analyzing the greedy algorithm is that there can be complex dependencies between a greedy match that occurred at some time step in some region of the line and the set of remaining servers that are available at a later time step in a completely different region of the line. In other words, a single greedy match at some time step can have a butterfly effect on the servers that will be available in the future in different regions. Algorithms such as hierarchical greedy that partition the interval in different regions have been designed to prevent matching decisions in one region from impacting the future available servers in another region. This does not necessarily lead to algorithms that are better than greedy but does give algorithms that are simpler to analyze.

A high-level contribution of our paper is to develop a general framework for analyzing the greedy algorithm, for both upper and lower bounds, that, we believe, also provides foundations for analyzing greedy in higher dimensions and other partially random models. The starting point of

our analysis is to consider a *hybrid algorithm* $\mathcal{H}^m_{\mathcal{A}}$ that matches the first m requests according to an algorithm $\mathcal{A}$ and then greedily matches each of the remaining requests to the closest available server. The algorithm $\mathcal{A}$ is different for each of our results. To derive our upper bound results, we first show a *hybrid lemma* that upper bounds, for any algorithm $\mathcal{A}$ that satisfies some fairly general properties, the difference $cost(\mathcal{H}^{m-1}_{\mathcal{A}}) - cost(\mathcal{H}^m_{\mathcal{A}})$ (i.e., between the total costs incurred by $\mathcal{H}^{m-1}_{\mathcal{A}}$ and $\mathcal{H}^m_{\mathcal{A}}$) as a function of the cost incurred by $\mathcal{A}$ to match the m^{th} request. This hybrid algorithm idea was also used in [Gupta and Lewi, 2012] to show a $O(\log(n))$ upper bound on the competitive ratio of a randomized greedy algorithm for online matching, but with three main differences. The first is that their hybrid algorithm is used to analyze a randomized algorithm on a deterministic instance (instead of a deterministic algorithm on a randomized instance). The second is that their hybrid algorithm uses an optimal offline algorithm $\mathcal{A}$, which we cannot use because we need to exploit the randomness of the instance, so we instead use existing online algorithms. The third is that our bound on $cost(\mathcal{H}^{m-1}_{\mathcal{A}}) - cost(\mathcal{H}^m_{\mathcal{A}})$ is tighter, which was a necessary improvement to obtain a constant competitive ratio in the fully random model.

The second part of the analysis of the upper bounds leverages the hybrid lemma. For the fully random model, we consider the hybrid algorithm $\mathcal{H}^m_{\mathcal{A}}$ where $\mathcal{A}$ is the constant-competitive hierarchical greedy algorithm by Kanoria [2022]. We note that a direct application of the hybrid lemma with this hybrid algorithm would only give an $O(\log n)$ competitive ratio for greedy. Instead, we also show that the total cost of the hierarchical greedy algorithm $\mathcal{A}$ is dominated by the cost of requests that are matched to servers at a constant distance away, which is needed to show that the difference between the costs of greedy and hierarchical greedy is $O(\sqrt{n})$. Since the expected optimal total cost is known to be $\Theta(\sqrt{n})$ and hierarchical greedy is constant competitive, we get that greedy is also constant competitive. For the random requests model, we again use the hybrid lemma but with a different algorithm $\mathcal{A}$, which is a simple modification of the fair-bias algorithm by Gupta et al. [2019], to show that greedy achieves an $O(\log n)$ competitive ratio.

For the $\Omega(\log n)$ lower bound in the random requests model, we consider an instance where there is a large number of servers at location 0, no servers in $(0, n^{-1/5}]$, and the remaining $1 - o(1)$ servers uniformly spread in $(n^{-1/5}, 1]$. We again analyze the difference $cost(\mathcal{H}^{m-1}_{\mathcal{A}}) - cost(\mathcal{H}^m_{\mathcal{A}})$, but where $\mathcal{A}$ is the tailored algorithm that matches any request in $[0, n^{-1/5}]$ to a server at 0 and greedily matches any other request to the closest available server. We show that at any time step t, the set of available servers for $\mathcal{H}^{m-1}_{\mathcal{A}}$ and $\mathcal{H}^m_{\mathcal{A}}$ differ in at most one server. We then consider the distance δ_t at time t between these two different servers that are available to only one of the algorithms and we show that $cost(\mathcal{H}^{m-1}_{\mathcal{A}}) - cost(\mathcal{H}^m_{\mathcal{A}})$ can be lower bounded as a function of $\max_{t \geq m} \delta_t$. Due to the randomness of the requests, the main difficulty is to lower bound $\max_{t \geq m} \delta_t$ (e.g., the gap δ_t can either shrink or expand at each time step), which we do by giving a careful partial characterization of the remaining servers $(S_0, \ldots, S_n)$ for $\mathcal{H}^m_{\mathcal{A}}$ at each time t that allows to analyze $(S_0, \ldots, S_n)$ and $(\delta_0, \ldots, \delta_n)$ separately.

1.2 Additional related work

In general metric spaces with adversarial requests and servers, Kalyanasundaram and Pruhs [1993] gave a $2n - 1$ deterministic competitive algorithm and proved that this competitive ratio is optimal for deterministic algorithms. On the line, Kalyanasundaram and Pruhs [1993] showed that the competitive ratio of greedy is at least $2^n - 1$. A deterministic algorithm with a sublinear competitive ratio was presented in [Antoniadis et al., 2014]. A few years later, Nayyar and Raghvendra [2017] gave a $O(\log^2 n)$ competitive deterministic algorithm, which was then shown to be $O(\log n)$-competitive in [Raghvendra, 2018]. Regarding lower bounds, Fuchs et al. [2003] showed that no deterministic algorithm can achieve a competitive ratio strictly less than 9.001 on the line.

For randomized algorithms, still for adversarial requests and servers, Meyerson et al. [2006] and Csaba and Pluhár [2007] obtained a $O(\log^3 n)$ competitive ratio in general metric spaces using randomized tree embeddings, which was later improved to $O(\log^2 n)$ by Bansal et al. [2007]. On the line, and for doubling metrics, Gupta and Lewi [2012] showed that a randomized greedy algorithm is $O(\log n)$ competitive. Recently, Peserico and Scquizzato [2021] improved the lower bound from [Fuchs et al., 2003] to obtain an $\Omega(\sqrt{\log n})$ lower bound for the line that also holds for randomized algorithms. For general metrics, it was previously known that no randomized algorithm can achieve a competitive ratio better than $\Omega(\log n)$ [Meyerson et al., 2006].

When the arrival order of the requests is random, Gairing and Klimm [2019] showed that greedy is $O(n)$ and $\Omega(n^{0.22})$ competitive. Raghvendra [2016] gave a deterministic algorithm that achieves a $O(\log n)$ competitive ratio, which is optimal even for randomized algorithms. When the requests are drawn i.i.d. from any distribution over the set of servers, Gupta et al. [2019] gave a $O((\log\log\log n)^2)$ competitive algorithm in general metric spaces that is also constant competitive on the line and for tree metrics. When the servers and requests are uniformly and independently distributed, Tsai et al. [1994] showed that greedy achieves an $2.3\sqrt{n}$ competitive ratio on the unit disk and Kanoria [2022] showed that an algorithm called hierarchical greedy is constant competitive on the unit hypercube (and also analyzed the more challenging fully dynamic setting where the servers also arrive online).

Empirical evaluations of different algorithms on real spatial data have shown that greedy performs well in practice [Tong et al., 2016]. The excess supply setting was studied by Akbarpour et al. [2022], who showed that the total optimal cost is $O(1)$ and the total cost of greedy is $O(\log^3 n)$ when the number of excess servers is linear and when the requests and servers are random (but the arrival order can be adversarial). The results for hierarchical greedy from [Kanoria, 2022] also extends to the excess supply setting. Kalyanasundaram and Pruhs [2000] showed a $O(\min(m, \log(n)))$ bound on the "double-competitive ratio" of greedy in an adversarial model with resource augmentation where there are m possible server locations and the adversary has only half as many servers at each location as greedy. Recourse, i.e. allowing matching decisions to be revoked to some extent, has been considered in [Gupta et al., 2020, Megow and Nölke, 2020]. In the offline non-bipartite version of the problem with $2n$ point drawn uniformly from $[0, 1]$, Frieze et al. [1990] showed that greedy achieves a $\Theta(\log n)$ approximation.

2 Preliminaries

In the online matching on the line problem, there are n_s *servers* $S = \{s_1, \ldots, s_{n_s}\}$ and $n = n_r$ *requests* $R = (r_1, \ldots, r_n)$ such that $s_i, r_i \in [0, 1]$ for all i. Hence, an instance is given by a pair (S, R). The servers are known to the algorithm at time $t = 0$. At each time step $t \in [n]$, the algorithm observes request r_t and must irrevocably match it to a server that has not yet been matched. We denote by $s_{\mathcal{A}}(r_t)$ the server that gets matched to request r_t by (the current execution of) algorithm $\mathcal{A}$ and by $S_{\mathcal{A},0} \supseteq \cdots \supseteq S_{\mathcal{A},n}$ the sets of free servers obtained through the execution of $\mathcal{A}$, where $S_{\mathcal{A},0}$ is the initial set of servers, and for all $t \in [n]$, $S_{\mathcal{A},t}$ is the set of remaining free servers just after matching r_t. The cost incurred from matching r_t to $s_{\mathcal{A}}(r_t)$ is $\text{cost}_t(\mathcal{A}, r_t) = |r_t - s_{\mathcal{A}}(r_t)|$ and the total cost of the matching produced by $\mathcal{A}$ on instance I is $\text{cost}(\mathcal{A}, I) = \sum_{t=1}^{n} \text{cost}_t(\mathcal{A}, r_t)$. We often abuse notation and write $\text{cost}_t(\mathcal{A})$, $\text{cost}(\mathcal{A})$, and S_t instead of $\text{cost}_t(\mathcal{A}, r_t)$, $\text{cost}(\mathcal{A}, I)$, and $S_{\mathcal{A},t}$.

All models studied in the paper can be represented by a triple (n^u, n^d, n). Here, n^u (resp. n) is the cardinality of the set S^u of servers (resp. of the set R of requests) sampled independently from the uniform distribution $\mathcal{U}_{[0,1]}$. n^d is the number of adversarily placed servers (hence, $n^u + n^d = n_s$). The performance of an algorithm $\mathcal{A}$ is measured by its competitive ratio:

$$\max_{S^d \in [0,1]^{n^d}} \frac{\mathbb{E}_{S^u,R\sim\mathcal{U}_{[0,1]},\mathcal{A}}[\text{cost}(\mathcal{A},(S^d \cup S^u,R))]}{\mathbb{E}_{S^u,R\sim\mathcal{U}_{[0,1]}}[\text{cost}(OPT,(S^d \cup S^u,R))]}.$$

where OPT is the offline optimal matching when the requests are known at time $t = 0$. We say that an algorithm is α-competitive if its competitive ratio is upper bounded by α. Although some papers in online optimization use a different notion of competitive ratio (see, e.g., survey [Mehta, 2013]), in the context of online matching on the line, most literature we are aware of use the same definition as ours. This is true, in particular, for papers over which we build [Gairing and Klimm, 2019, Gupta et al., 2019, Kanoria, 2022] or whose results we improve [Akbarpour et al., 2022, Tsai et al., 1994].

The three models investigated in this paper can then be formalized as follows.

- In the *fully random model*, $(n^u, n^d, n) = (n, 0, n)$, i.e., all servers S and requests R are drawn uniformly and independently from $[0, 1]$ and there is an equal number of servers and requests.
- For a constant $\epsilon > 0$, we define the *fully random $\epsilon-$excess model*, in which $(n^u, n^d, n) = ((1+\epsilon)n, 0, n)$, i.e., all servers S and requests R are drawn uniformly and independently from $[0, 1]$ and there is a linear excess of ϵn servers.
- In the *random requests model*, $(n^u, n^d, n) = (0, n, n)$, i.e., the requests R are still drawn uniformly and independently from $[0, 1]$ but the servers are now chosen adversarially over all potential sequence of n requests in $[0, 1]$.

The greedy algorithm, denoted by $\mathcal{G}$, is the algorithm that matches each request r_t to the closest available server, i.e., $s_{\mathcal{G}}(r_t) = \arg\min_{s\in S_{\mathcal{G},t-1}} |s - r_t|$. We say that an algorithm $\mathcal{A}$ *makes neighboring matches* if it matches every request r_t either to the closest available server to its left or to its right. For any algorithm $\mathcal{A}$ (possibly randomized) and $m \in \{0, \ldots, n\}$, we define the hybrid algorithm $\mathcal{H}^m_{\mathcal{A}}$ that matches the first m requests according to $\mathcal{A}$ and then greedily matches the remaining requests to the closest available server. The following key lemma (proved in the full version of the paper) bounds $\mathbb{E}\big[cost(\mathcal{H}^{m-1}_{\mathcal{A}}) - cost(\mathcal{H}^m_{\mathcal{A}})\big]$ as a function of $\text{cost}_m(\mathcal{A})$ – that is, the cost for algorithm $\mathcal{A}$ to match the m^{th} request.

Lemma 2.1. ***(The Hybrid Lemma).*** *There exists a constant $C > 0$ such that for any online algorithm $\mathcal{A}$ that makes neighboring matches, for any instance with n arbitrary servers $S = \{s_1, \ldots, s_n\}$, n requests $R = (r_1, \ldots, r_n)$ uniformly and independently drawn from $[0, 1]$, for any $m \in [n]$, we have*

$$\mathbb{E}\big[cost(\mathcal{H}^{m-1}_{\mathcal{A}}) - cost(\mathcal{H}^m_{\mathcal{A}})|S_{m-1}, r_m\big] \le C \cdot \mathbb{E}\left[\left(1 + \log\left(\tfrac{1}{cost_m(\mathcal{A})}\right)\right) cost_m(\mathcal{A})\middle| S_{m-1}, r_m\right].$$

Note that the expectation is taken over the randomness in the requests sequence as well as any possible source of randomization in the algorithm $\mathcal{A}$. The idea of using hybrid algorithms for analyzing online matching algorithms was used in [Gupta and Lewi, 2012], who also introduce a hybrid lemma (see Section 1.1 for additional discussion). A key component of the proof of Lemma 2.1 relies on Lemma 2.2 given below, that describes, for a fixed $m \in [n]$, the difference between the executions of $\mathcal{H}^m_{\mathcal{A}}$ and $\mathcal{H}^{m-1}_{\mathcal{A}}$ on the same sequence R. Lemma 2.2 in fact shows that, at every step t (i.e., just after matching request r_t), the free servers for both algorithms coincide, with the exception of at most a pair of servers, that we denote by $g^L_t < g^R_t$ (see Figure 1) ; there is no other free server in between g^L_t and g^R_t ; and that strong bounds can be obtained on $\delta_t := g^L_t - g^R_t$. These properties, in turns, will allow us to control the difference in the costs incurred by the two algorithms, eventually leading to the bound from Lemma 2.1.

To ease the exposition, we drop the reference to the algorithms in the indices and write S_t and $s(r_t)$ instead of $S_{\mathcal{H}^m_{\mathcal{A}},t}$ and $s_{\mathcal{H}^m_{\mathcal{A}}}(r_t)$ to denote, respectively, the set of free server for $\mathcal{H}^m_{\mathcal{A}}$ just after

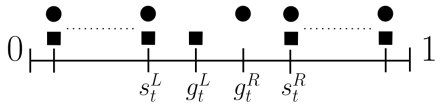

Fig. 1. Set of servers S_t (free servers at time t for $\mathcal{H}_A^m$) and S'_t (free servers at time t for $\mathcal{H}_A^{m-1}$) in the case where $S_t \neq S'_t$, where the squares are the servers in S_t and the circles the servers in S'_t.

matching r_t and the server to which $\mathcal{H}_{\mathcal{A}}^m$ matches r_t. Similarly, we write S'_t and $s'(r_t)$ instead of $S_{\mathcal{H}_{\mathcal{A}}^{m-1},t}$ and $s_{\mathcal{H}_{\mathcal{A}}^{m-1}}(r_t)$ for the equivalent objects for $\mathcal{H}_{\mathcal{A}}^{m-1}$.

If $S_t = S'_t$, then we write $g_t^L = g_t^R = \emptyset$ and $\delta_t = 0$. We also define $s_t^L = \max\{s \in S_t \cup S'_t \setminus \{g_t^L, g_t^R\} : s \leq g_t^L\}$ and $s_t^R = \min\{s \in S_t \cup S'_t \setminus \{g_t^L, g_t^R\} : s \geq g_t^R\}$ (with the convention that $s_t^L = \emptyset$ if $\{S_t \cup S'_t \setminus \{g_t^L, g_t^R\} : s \leq g_t^L\} = \emptyset$ or if $g_t^L = \emptyset$, and similarly for s_t^R), which are the nearest servers of S_t (or equivalently, of S'_t) on the left of g_t^L and on the right of g_t^R.

Lemma 2.2. *Let $\mathcal{A}$ be any online algorithm that makes neighboring matches, S_0 be n arbitrary servers and R be n arbitrary requests. Let $(S_0, \ldots, S_n)$ and $(S'_0, \ldots, S'_n)$ denote the set of free servers for $\mathcal{H}_{\mathcal{A}}^m$ and $\mathcal{H}_{\mathcal{A}}^{m-1}$ at each time steps. Then, the following propositions hold for all $t \in \{m, \ldots, n\}$:*

(1) ***Difference in at most one server.*** *$|S_t \setminus S'_t| = |S'_t \setminus S_t| \leq 1$.*

(2) ***Consecutiveness of the different servers.*** *If $g_t^L, g_t^R \neq \emptyset$, there is no server $s \in S_t \cup S'_t$ such that $g_t^L < s < g_t^R$.*

(3) ***Gap remains zero after disappearing.*** *If $\delta_t = 0$, then $\delta_{t'} = 0$ for all $t' \geq t$.*

The proof is given in the full version of the paper. For all $t < n$ and $S_t \neq S'_t$, we also characterize the values of $s(r_{t+1}), s'(r_{t+1}), \delta_{t+1}, g_{t+1}^L, g_{t+1}^R$, and give an upper bound on $\Delta\text{cost}_{t+1} := |\text{cost}_{t+1}(\mathcal{H}^{m-1}) - \text{cost}_{t+1}(\mathcal{H}^m)|$ (see the full version of the paper).

3 Greedy is Constant Competitive in the Fully Random Model

In this section, we show that greedy achieves a constant competitive ratio in the fully random model where both the servers and requests are drawn uniformly and independently from $[0, 1]$. In addition, we show that this result also holds when there is a linear excess supply of servers.

The setting with n servers. We recall that in this setting, the competitive ratio of any algorithm $\mathcal{A}$ is given by:

$$\frac{\mathbb{E}_{(R,S)\sim\mathcal{U}_{(0,1)^n}\times\mathcal{U}_{(0,1)^n},\mathcal{A}}[\text{cost}(\mathcal{A},(S,R))]}{\mathbb{E}_{(R,S)\sim\mathcal{U}_{(0,1)^n}\times\mathcal{U}_{(0,1)^n}}[\text{cost}(OPT,(S,R))]}.$$

The main idea of the analysis is to consider a hybrid algorithm that first runs the hierarchical greedy algorithm from [Kanoria, 2022], and then greedily matches the remaining requests to the closest available server.

We first present the hierarchical greedy algorithm introduced in [Kanoria, 2022], which we denote by $\mathcal{A}^H$ (note that [Kanoria, 2022] considers two models: a *semi-dynamic* model similar to ours, and a *fully-dynamic* model where the servers also arrive online. We only present here the algorithm corresponding to the *semi-dynamic* model). To describe it, we need to define the sequence $\mathcal{I}_{\ell_0}, ..., \mathcal{I}_0$, where $\ell_0 = \log(n)$, which are increasingly refined partitions of $[0, 1]$. More precisely, $\mathcal{I}_{\ell_0} = \{[0, 1]\}$ and for each $\ell \leq \ell_0 - 1$, $\mathcal{I}_\ell$ is the partition obtained by dividing each interval in $\mathcal{I}_{\ell+1}$ into two intervals of equal length, i.e., $\mathcal{I}_\ell = \left(\cup_{[0,y]\in\mathcal{I}_{\ell+1}}\{[0, y/2],]y/2, y]\}\right) \cup \left(\cup_{]x,y]\in\mathcal{I}_{\ell+1}}\{]x, (x+y)/2],](x+y)/2, y]\}\right)$. The partitions obtained through this process can be organized in a binary tree, where the nodes at level ℓ are the intervals of $\mathcal{I}_\ell$ and the leafs are the intervals of $\mathcal{I}_0$.

Given a request r_t, let $I(r_t)$ be the leaf interval to which r_t belongs and $J(r_t)$ be the lowest-level ancestor interval of $I(r_t)$ in the tree such that $J(r_t) \cap S_{t-1} \neq \emptyset$, i.e., such that $J(r_t)$ contains some free servers when request r_t arrives. The hierarchical greedy algorithm matches r_t to any free server in $J(r_t)$. For our purposes, we assume that it matches r_t to the closest free server in $J(r_t)$. A request r_t is said to be *matched at level* ℓ if $J(r_t) \in \mathcal{I}_\ell$. There are two known results about hierarchical greedy that are important for our analysis. The first one upper bounds the number of requests matched at each level.

Lemma 3.1 ([Kanoria, 2022]). *There is a constant $C' > 0$ such that, for all $\ell \in \{0, \dots, \ell_0\}$, we have $\mathbb{E}[|\{r_t : J(r_t) \in \mathcal{I}_\ell\}|] \leq C'\sqrt{n2^{\ell-\ell_0}}2^{\ell_0-\ell}$.*

The second important result about hierarchical greedy is its constant competitiveness

Theorem 3.2 ([Kanoria, 2022]). *In the fully random model, we have that $\mathbb{E}[cost(\mathcal{A}^H)] = O(\sqrt{n})$.*

Next, we show the following bound on the cost incurred by hierarchical greedy when matching a request at level ℓ.

Lemma 3.3. *For all $t \in [n]$, if r_t is matched at level ℓ, then we have*

$$cost_t(\mathcal{A}^H)\log(1/cost_t(\mathcal{A}^H)) \leq 2^{\ell-\ell_0}(\log(2)(\ell_0-\ell)+1).$$

Proof. Let $\ell \in \{0, \dots, \ell_0\}$. First note that the cost incurred by $\mathcal{A}^H$ when matching a request r_t at level ℓ satisfies $\text{cost}_t(\mathcal{A}^H) \leq 2^{\ell-\ell_0}$ since the intervals of $\mathcal{I}_\ell$ have length at most $2^{\ell-\ell_0}$ by definition of $\mathcal{I}_\ell$. Next, if $2^{\ell-\ell_0} \in (0, 1/e]$, then

$$\text{cost}_t(\mathcal{A}^H)\log(1/\text{cost}_t(\mathcal{A}^H)) \leq 2^{\ell-\ell_0}\log(1/2^{\ell-\ell_0})$$

since $x\log(1/x)$ is non-decreasing on $(0, 1/e]$ and $\text{cost}_t(\mathcal{A}^H) \leq 2^{\ell-\ell_0}$. If $2^{\ell-\ell_0} \in [1/e, 1]$, then

$$\text{cost}_t(\mathcal{A}^H)\log(1/\text{cost}_t(\mathcal{A}^H)) \leq 1/e \leq 2^{\ell-\ell_0}$$

since $\arg\max_{x\in(0,1]} x\log(1/x) = 1/e$ and $\frac{1}{e}\log(\frac{1}{1/e}) = \frac{1}{e}$. We conclude that if r_t is matched at level ℓ,

$$\text{cost}_t(\mathcal{A}^H)\log(1/\text{cost}_t(\mathcal{A}^H)) \leq 2^{\ell-\ell_0}\log(1/2^{\ell-\ell_0}) + 2^{\ell-\ell_0} \leq 2^{\ell-\ell_0}(\log(2)(\ell_0-\ell)+1). \qquad \square$$

The next lemma is the main lemma of this section and shows that the difference between the total cost of greedy and hierarchical greedy is $O(\sqrt{n})$.

Lemma 3.4. *In the fully random model, we have that*

$$\mathbb{E}[cost(\mathcal{G}) - cost(\mathcal{A}^H)] = O(\sqrt{n}).$$

Proof. We first note that since the hierarchical greedy algorithm matches every request r_t to the closest free server in $J(r_t)$, and since $r_t \in J(r_t)$ by definition of $J(r_t)$, hierarchical greedy makes neighboring matches, which is the condition needed to apply the hybrid lemma to the hybrid algorithm $\mathcal{H}^m$. We get that

$$\begin{aligned}
&\mathbb{E}[cost(\mathcal{G}) - cost(\mathcal{A}^H)] \\
&= \sum_{m=1}^{n} \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m)] && \mathcal{H}^n = \mathcal{A}^H, \mathcal{H}^0 = \mathcal{G} \\
&\leq C\sum_{m=1}^{n} \mathbb{E}[\left(1 + \log\left(\tfrac{1}{cost_m(\mathcal{A}^H)}\right)\right)\text{cost}_m(\mathcal{A}^H)] && \text{Hybrid lemma}
\end{aligned}$$

$$\leq C \sum_{m=1}^{n} \mathbb{E}[\log\left(\tfrac{1}{cost_m(\mathcal{A}^H)}\right)\text{cost}_m(\mathcal{A}^H)] + C\mathbb{E}[cost(\mathcal{A}^H)]$$

$$\leq C \sum_{m=1}^{n} \mathbb{E}[\log\left(\tfrac{1}{cost_m(\mathcal{A}^H)}\right)\text{cost}_m(\mathcal{A}^H)] + O(\sqrt{n}) \qquad \text{Theorem 3.2}$$

$$= C \sum_{m=1}^{n} \sum_{\ell=0}^{\ell_0} \mathbb{P}(J(r_m) \in \mathcal{I}_\ell)\mathbb{E}[\log\left(\tfrac{1}{cost_m(\mathcal{A}^H)}\right)\text{cost}_m(\mathcal{A}^H)|J(r_m) \in \mathcal{I}_\ell] + O(\sqrt{n})$$

$$\leq C \sum_{m=1}^{n} \sum_{\ell=0}^{\ell_0} \mathbb{P}(J(r_m) \in \mathcal{I}_\ell) \cdot 2^{\ell-\ell_0}(\log(2)(\ell_0 - \ell) + 1) + O(\sqrt{n}) \qquad \text{Lemma 3.3}$$

$$= C \sum_{\ell=0}^{\ell_0} 2^{\ell-\ell_0}(\log(2)(\ell_0 - \ell) + 1) \cdot \sum_{m=1}^{n} \mathbb{P}(J(r_m) \in \mathcal{I}_\ell) + O(\sqrt{n})$$

$$= C \sum_{\ell=0}^{\ell_0} 2^{\ell-\ell_0}(\log(2)(\ell_0 - \ell) + 1) \cdot \mathbb{E}[|\{r_t : J(r_t) \in I_\ell\}|] + O(\sqrt{n})$$

$$\leq CC'\sqrt{n} \sum_{\ell=0}^{\ell_0} 2^{(\ell-\ell_0)/2}(\log(2)(\ell_0 - \ell) + 1) + O(\sqrt{n}) \qquad \text{Lemma 3.1}$$

$$= CC'\sqrt{n} \sum_{j=0}^{\ell_0} 2^{-j/2}(\log(2)j + 1) + O(\sqrt{n})$$

$$= CC'\sqrt{n}\left(\log(2) \sum_{j=0}^{\ell_0} j\left(\frac{1}{\sqrt{2}}\right)^j + \sum_{j=0}^{\ell_0}\left(\frac{1}{\sqrt{2}}\right)^j\right) + O(\sqrt{n})$$

$$= O(\sqrt{n}). \qquad \square$$

The last result needed is that the optimal cost in the fully random model is known to be $\Theta(\sqrt{n})$.

Lemma 3.5 ([Kanoria, 2022]). *In the fully random model, we have that* $\mathbb{E}[\texttt{OPT}] = \Theta(\sqrt{n})$.

By combining Theorem 3.2, Lemma 3.4, and Lemma 3.5, we obtain the main result of this section.

Theorem 1.1. *For online matching on the line in the fully random model, the greedy algorithm achieves a constant competitive ratio.*

The excess supply setting. We consider here an extension of the previous model where there is a linear excess of servers. For any constant $\epsilon > 0$, we define the *fully random ϵ-excess model*, where an instance consist of n requests and $n(1+\epsilon)$ servers all drawn uniformly and independently from $[0, 1]$. The competitive ratio of any algorithm $\mathcal{A}$ is given by:

$$\frac{\mathbb{E}_{(R,S)\sim\mathcal{U}(0,1)^n\times\mathcal{U}(0,1)^{n(1+\epsilon)},\mathcal{A}}[\text{cost}(\mathcal{A}, (S, R))]}{\mathbb{E}_{(R,S)\sim\mathcal{U}(0,1)^n\times\mathcal{U}(0,1)^{n(1+\epsilon)}}[\text{cost}(OPT, (S, R))]}.$$

In this setting, the hybrid approach with hierarchical greedy used above does not give a constant competitive ratio. However, we are still able to prove that greedy is constant competitive with a different argument. Unlike the model with n servers, the analysis for the excess supply setting does not rely on the hybrid lemma but on concentration arguments. Missing proofs can be found in the full version of the paper.

The main technical contribution here lies in showing that, thanks to the excess of servers, there is an exponentially small probability that there is a large area around the n-th request that contains no available servers. More formally, for $\ell, m \in [0,1]$, we let $x_{(\ell,m)} = |\{t \in [n-1] : r_t \in (\ell, m)\}|$ be the number of requests out of the $n-1$ first requests that arrived in the interval (ℓ, m), and we let $y_{(\ell,m)} = |\{t \in [n(1+\epsilon)] : s_t \in (\ell, m)\}|$ be the total number of servers that lie in the interval (ℓ, m). Then, the following lemma holds.

Lemma 3.6. *Let $\epsilon > 0$ be a constant. There are constants C_ϵ, C'_ϵ such that, in the fully random ϵ-excess model, we have that for all $z \in [\frac{4+\epsilon}{\epsilon n}, 1]$,*

$$\mathbb{P}(\exists \ell, m \in [0,1] : x_{(\ell,m)} = y_{(\ell,m)}, (r_n - \ell \geq z \text{ or } \ell = 0), (m - r_n \geq z \text{ or } m = 1) \mid r_n) \leq C'_\epsilon e^{-nzC_\epsilon}.$$

Using Lemma 3.6, we then upper bound the expected cost incurred by greedy at the last step. At a high level, we use in the proof that the free servers at each time step act as "natural barriers" between different areas of the interval $[0,1]$ (in the sense that if there is a free server at location $x \in [0,1]$, no request arriving in $[0,x]$ can be matched to a server in $(x,1]$, and vice-versa). This allows to quantify precisely the total number of remaining servers in each of those areas. Note that in [Kanoria, 2022], the analysis also relies on a division of space into distinct regions, and on a quantification of remaining servers and requests in each region. However, in [Kanoria, 2022], the division is fixed at the beginning of the time horizon (through the partition $\mathcal{I}_{\ell_0}, ..., \mathcal{I}_0$). The additional difficulty in our setting is that the "barriers" we consider depend on all previously arrived requests and are thus random.

Lemma 3.7. *Let $\epsilon > 0$ be a constant. There is a constant C''_ϵ such that, in the fully random ϵ-excess model, we have $\mathbb{E}[cost_n(\mathcal{G})] \leq \frac{C''_\epsilon}{n}$.*

Proof. To exclude any ambiguity, we condition on the event that all servers are distinct and that no server or requests are at positions 0 and 1, which occurs almost surely. In the remainder of the proof, we condition on the variable r_n and let $s_n^L = \max\{s \in S_{n-1} : s \leq r_n\}$ and $s_n^R = \min\{s \in S_{n-1} : s \geq r_n\}$ denote the nearest available servers on the left and on the right of r_n when r_n arrives; with the convention that $s_n^L = 0$ and $s_n^L = 1$ if there are no such servers.

Now, let $z \in [\frac{4(1+\epsilon/4)}{\epsilon n}, 1]$ and assume that $\text{cost}_n(\mathcal{G}) \geq z$. Since $\mathcal{G}$ matches r_n to the closest available server, we must have $r_n - s_n^L \geq z$ or $s_n^L = 0$, and $s_n^R - r_n \geq z$ or $s_n^R = 1$. In addition, by definition of s_n^L and s_n^R, we have that $(s_n^L, s_n^R) \cap S_{n-1} = \emptyset$. Now, recall that all requests $r_1, \ldots, r_{n-1}$ have been matched each time to the closest available server. Moreover, for all $j \in [n-1]$, s_n^L was either available when r_j arrives, but r_j was not matched to it, or $s_n^L = 0$; similarly for s_n^R. Hence, if $r_j \notin (s_n^L, s_n^R)$, then $s_{\mathcal{G}}(r_j) \notin (s_n^L, s_n^R)$. Similarly, if $r_j \in (s_n^L, s_n^R)$, then $s_{\mathcal{G}}(r_j) \in (s_n^L, s_n^R)$. Therefore,

$$|\{j \in [n-1] : s_{\mathcal{G}}(r_j) \in (s_n^L, s_n^R)\}| = |\{j \in [n-1] : r_j \in (s_n^L, s_n^R)\}|.$$

In addition, since $(s_n^L, s_n^R) \cap S_{n-1} = \emptyset$, all servers in $(s_n^L, s_n^R) \cap S_0$ must have been matched to some request before time $n-1$, hence

$$|\{j \in [n-1] : s_{\mathcal{G}}(r_j) \in (s_n^L, s_n^R)\}| = |\{j \in [n(1+\epsilon)] : s_j \in (s_n^L, s_n^R)\}|.$$

By combining the two previous equalities and by definition of $x_{(s_n^L, s_n^R)}$, and $y_{(s_n^L, s_n^R)}$, we get that

$$x_{(s_n^L, s_n^R)} = |\{j \in [n-1] : r_j \in (s_n^L, s_n^R)\}| = |\{j \in [n(1+\epsilon)] : s_j \in (s_n^L, s_n^R)\}| = y_{(s_n^L, s_n^R)}.$$

Since we have that $r_n - s_n^L \geq z$ or $s_n^L = 0$, and $s_n^R - r_n \geq z$ or $s_n^R = 1$, we thus have that

$$\begin{aligned}&\mathbb{P}(\text{cost}_n(\mathcal{G}) \geq z \mid r_n)\\ &\quad\leq \mathbb{P}(\exists \ell, m \in [0,1] : x_{(\ell,m)} = y_{(\ell,m)}, (r_n - \ell \geq z \text{ or } \ell = 0), (m - r_n \geq z \text{ or } m = 1) \mid r_n). \quad (1)\end{aligned}$$

Now, by Lemma 3.6, we have that for some constants $C_\epsilon, C'_\epsilon > 0$:

$$\mathbb{P}(\exists \ell, m \in [0,1] : x_{(\ell,m)} = y_{(\ell,m)}, (r_n - \ell \geq z \text{ or } \ell = 0), (m - r_n \geq z \text{ or } m = 1) \mid r_n) \leq C'_\epsilon e^{-nzC_\epsilon}.$$

Combining this with (1) and by the law of total probability, we get $\mathbb{P}(\text{cost}_n(\mathcal{G}) \geq z) \leq C'_\epsilon e^{-nzC_\epsilon}$. Hence, we obtain

$$\begin{aligned}
\mathbb{E}[cost_n(\mathcal{G})] &\leq \tfrac{4(1+\epsilon/4)}{\epsilon n} + \int_{z=\frac{4(1+\epsilon/4)}{\epsilon n}}^{1} \mathbb{P}(cost_n(\mathcal{G}) \geq z)\mathrm{d}z \\
&\leq \tfrac{4(1+\epsilon/4)}{\epsilon n} + \int_{z=\frac{4(1+\epsilon/4)}{\epsilon n}}^{1} C'_\epsilon e^{-nzC_\epsilon}\mathrm{d}z \\
&\leq \tfrac{4(1+\epsilon/4)}{\epsilon n} + \frac{C'_\epsilon}{C_\epsilon n}. && \text{for some } C_\epsilon > 0 \\
&= \frac{C''_\epsilon}{n} && \text{for some } C''_\epsilon > 0 \qquad \square
\end{aligned}$$

We underscore that a simple application of Chernoff bounds between all initial pairs of servers locations would only lead to a weaker version of the above lemma, involving poly-logarithmic terms. Since our objective was to present a sharp analysis of greedy, we introduced the refined analysis above.

Last, we observe that, because of servers getting less and less dense as requests arrive, the expected cost at each step of the greedy algorithm increases.

Lemma 3.8. *Let $\epsilon > 0$ be a constant. Then, in the fully random ϵ-excess model, we have that for all $i \in [n-1]$, $\mathbb{E}[cost_i(\mathcal{G})] \leq \mathbb{E}[cost_{i+1}(\mathcal{G})]$.*

Using Lemma 3.7 and Lemma 3.8, we conclude that $\mathbb{E}[cost(\mathcal{G})] = \sum_{i=1}^{n} \mathbb{E}[cost_i(\mathcal{G})] \leq n \cdot \mathbb{E}[cost_n(\mathcal{G})] \leq C''_\epsilon$. We have thus shown the following.

Lemma 3.9. *Let $\epsilon > 0$ be a constant. There exists a constant $C''_\epsilon > 0$ such that in the fully random ϵ-excess model, we have $\mathbb{E}[cost(\mathcal{G})] \leq C''_\epsilon$.*

In order to conclude the proof of Theorem 1.2, it suffices to lower bound the cost of the optimal solution in the *fully random ϵ-excess model.*

Lemma 3.10 ([Kanoria, 2022]). *For any constant $\epsilon > 0$, we have that in the fully random ϵ-excess model, $\mathbb{E}[\mathtt{OPT}] = \Theta(\frac{1}{\epsilon})$.*

We can then conclude the following result on the performance of the greedy algorithm.

Theorem 1.2. *For any constant $\epsilon > 0$, greedy is $O(1)$-competitive in the fully random ϵ-excess model.*

4 Greedy is Logarithmic Competitive in the Random Requests Model

In this section, we show that greedy achieves an $\Theta(\log n)$ competitive ratio in the random requests model where the servers are chosen adversarially and the requests are drawn uniformly and independently from $[0,1]$. Thus, unlike in the fully random model, servers and requests can be distributed in a significantly different manner in this model.

4.1 Greedy is $O(\log n)$-competitive

We first show the $O(\log n)$ upper bound. We note that, even though hierarchical greedy and greedy are both constant-competitive in the fully random model, hierarchical greedy is only $\Omega(n^{1/4})$-competitive in the random requests model (see the full version of the paper). The main lemma

(Lemma 4.2) shows that greedy is at most a logarithmic factor away from any online algorithm that makes neighboring matches. To prove Lemma 4.2, we first need to lower bound the probability that the cost incurred by any online algorithm at any time step is small. First, for all $t \in [n]$, we let $\mathcal{G}(S_{t-1}) = \{r \in [0,1] : \exists s \in S_{t-1}, |r-s| < \frac{1}{n^4}\}$ be the set of points in $[0,1]$ that are close to servers in S_{t-1}.

Lemma 4.1. *In the random requests model, for any online algorithm $\mathcal{A}$ and any time step $t \in [n]$, we have that $\mathbb{E}[cost_t(\mathcal{A})] \geq \frac{1}{2(n+1)}$ and that $\mathbb{P}(r_t \in \mathcal{G}(S_{t-1})) \leq \frac{2}{n^3}$.*

The proof is in the full version of the paper. Next, to show that Lemma 4.2 holds for any online algorithm $\mathcal{A}$ that makes neighboring matches, we use the hybrid lemma on the hybrid algorithm $\mathcal{H}^m_{\mathcal{A}}$ (and we abuse notation by writing $\mathcal{H}^m$).

Lemma 4.2. *In the random requests model, there exists a constant $C > 0$ such that for any online algorithm $\mathcal{A}$ that makes neighboring matches,*

$$\mathbb{E}[cost(\mathcal{G})] \leq C \log(n) \mathbb{E}[cost(\mathcal{A})].$$

Proof. We write

$$\begin{aligned}
&\mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m)] \\
&= \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) \mid r_m \notin \mathcal{G}(S_{m-1})] \cdot \mathbb{P}(r_m \notin \mathcal{G}(S_{m-1})) \\
&\qquad + \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) \mid r_m \in \mathcal{G}(S_{m-1})] \cdot \mathbb{P}(r_m \in \mathcal{G}(S_{m-1})) \\
&\leq \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) \mid r_m \notin \mathcal{G}(S_{m-1})] \cdot \mathbb{P}(r_m \notin \mathcal{G}(S_{m-1})) + n \cdot 2/n^3 \\
&\leq C \cdot \mathbb{E}\big[\big(1 + \log\big(\tfrac{1}{cost_m(\mathcal{A})}\big)\big)\mathrm{cost}_m(\mathcal{A}) \mid r_m \notin \mathcal{G}(S_{m-1})\big] \cdot \mathbb{P}(r_m \notin \mathcal{G}(S_{m-1})) + 2n^{-2} \\
&\leq C(1 + 4\log(n)) \cdot \mathbb{E}[\mathrm{cost}_m(\mathcal{A}) \mid r_m \notin \mathcal{G}(S_{m-1})] \cdot \mathbb{P}(r_m \notin \mathcal{G}(S_{m-1})) + 2n^{-2} \\
&\leq C(1 + 4\log(n)) \cdot \mathbb{E}[\mathrm{cost}_m(\mathcal{A})] + 2n^{-2} \\
&= C' \log(n) \cdot \mathbb{E}[\mathrm{cost}_m(\mathcal{A})],
\end{aligned}$$

where the first inequality is by Lemma 4.1, the second one by the Hybrid Lemma (Lemma 2.1 ; noting that $\{r_m \notin \mathcal{G}(S_{m-1})\}$ is an event that depends only on S_{m-1} and r_m and that $\mathcal{A}$ makes neighboring matches) and the third one is since for any algorithm $\mathcal{A}$, $\mathrm{cost}_m(\mathcal{A}) \geq 1/n^4$ when $r_m \notin \mathcal{G}(S_{m-1})$. The last equality is by Lemma 4.1.

Since $\mathcal{H}^n = \mathcal{A}$ and $\mathcal{H}^0 = \mathcal{G}$, we conclude that

$$\begin{aligned}
\mathbb{E}[cost(\mathcal{G}) - cost(\mathcal{A})] &= \sum_{m=1}^{n} \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m)] \\
&\leq C' \log(n) \sum_{m=1}^{n} \mathbb{E}[cost_m(\mathcal{A})] \\
&= C' \log(n) \cdot \mathbb{E}[cost(\mathcal{A})]. \qquad \square
\end{aligned}$$

It remains to show the existence of a $O(1)$-competitive online algorithm that makes neighboring matches in the random requests model, which is the case for a simple modification of the algorithm fair-bias from [Gupta et al., 2019]. The proof is deferred to the full version of the paper.

Lemma 4.3. *In the random requests model, there exists a $O(1)$-competitive algorithm that makes neighboring matches.*

We are now ready to prove the main result of Section 4.1.

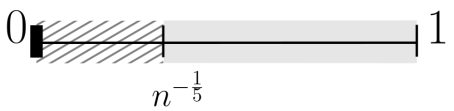

Fig. 2. The lower bound instance. There are $n^{4/5} + 4\log^2(n)\sqrt{n}$ servers at 0, no server in the dashed area, and $n - (n^{\frac{4}{5}} + 4\log^2(n)\sqrt{n})$ servers uniformly distributed in the gray area.

Theorem 1.3. *For online matching on the line in the random requests model, the greedy algorithm achieves an $O(\log n)$-competitive ratio.*

Proof. By Lemma 4.3, there exists an algorithm $\mathcal{A}$ that is $O(1)$-competitive algorithm in the random requests model and makes neighboring matches. We have, by Lemma 4.2, that $\mathbb{E}[cost(\mathcal{G})] \leq C\log(n)\mathbb{E}[cost(\mathcal{A})]$. We conclude that greedy is $O(\log n)$-competitive. □

4.2 Overview of the lower bound

The $\Omega(\log n)$ lower bound is the main technical proof of this paper. It is obtained by analyzing another hybrid algorithm to show that, on some instance, greedy makes mistakes that have an intricate cascading effect on the cost of future requests. In this section, we give an overview of the proof of the lower bound. The complete analysis and proofs of all lemmas can be found in the full version of the paper.

4.2.1 Description of the instance. We define the set of servers S_0 such that there are $n^{4/5} + 4\log(n)^2\sqrt{n}$ servers located at point 0, there are no servers in the interval $(0, n^{-1/5}]$ and the remaining $n - (n^{4/5} + 4\log(n)^2\sqrt{n})$ servers are uniformly spread in the interval $(n^{-1/5}, 1]$. More precisely, for all $j \in [n^{4/5}+4\log(n)^2\sqrt{n}]$, we set $s_j = 0$. Then, we let $\tilde{n} := n-4\log(n)^2\sqrt{n}/(1-n^{-1/5})$, and for all $j \in [n - (n^{4/5} - 4\log(n)^2\sqrt{n})]$, we set $s_{(n^{4/5}+4\log(n)^2\sqrt{n})+j} = n^{-1/5} + \frac{j}{\tilde{n}}$ (see Figure 2 for an illustration of the instance). We note that, interestingly, the servers are almost uniform since a $1 - o(1)$ fraction of the servers are uniformly spread in an interval $(o(1), 1]$.

4.2.2 Analysis of the instance. We compare the greedy algorithm to the algorithm $\mathcal{A}$ that, for all $t \in [n]$, matches r_t to a free server at location 0 if $r_t \in [0, n^{-1/5}]$ and $S_{\mathcal{A},t-1} \cap \{0\} \neq \emptyset$, and, otherwise, matches r_t greedily. Note that for the instance defined above, $\mathcal{A}$ is a better algorithm than greedy since the expected total number of requests in $[0, n^{-1/5}]$ is $n^{-1/5} \cdot n = n^{4/5}$, which is less than the number of servers at position 0. The main part of the proof is to lower bound $\mathbb{E}[cost(\mathcal{H}^{m-1}_{\mathcal{A}}) - cost(\mathcal{H}^{m}_{\mathcal{A}})]$, i.e., the increase in cost from switching from algorithm $\mathcal{A}$ to the greedy algorithm $\mathcal{G}$ one step earlier in hybrid algorithm $\mathcal{H}^{m-1}_{\mathcal{A}}$ compared to $\mathcal{H}^{m}_{\mathcal{A}}$. As we will show, matching a request in $[0, n^{-1/5}]$ greedily at time $t = m$ instead of matching it to a server at location 0 causes a cascading increase in costs at future time steps for $\mathcal{H}^{m-1}_{\mathcal{A}}$ compared to $\mathcal{H}^{m}_{\mathcal{A}}$ due to the different available servers, even though these two algorithms both match requests greedily at time steps $t > m$.

Structural properties. The first lemma shows that at every time step t, there are at most two servers in the symmetric difference between the sets of free servers $S_{\mathcal{H}^{m}_{\mathcal{A}},t}$ and $S_{\mathcal{H}^{m-1}_{\mathcal{A}},t}$, and that the potential extra free server in $S_{\mathcal{H}^{m-1}_{\mathcal{A}},t}$ is always located at 0 whereas the potential extra free server in $S_{\mathcal{H}^{m}_{\mathcal{A}},t}$ is the leftmost free server that is not at location 0 (see Figure 3). To ease notation, we write $\mathcal{H}^m$ and $\mathcal{H}^{m-1}$ instead of $\mathcal{H}^{m}_{\mathcal{A}}$ and $\mathcal{H}^{m-1}_{\mathcal{A}}$ and S_t and S'_t instead of $S_{\mathcal{H}^m,t}$ and $S_{\mathcal{H}^{m-1},t}$.

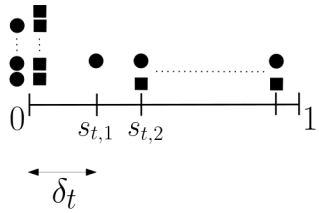

Fig. 3. Sets of free servers for $\mathcal{H}^m$ and $\mathcal{H}^{m-1}$ at all time steps (with the circles denoting servers in S_t and the squares denoting servers in S'_t).

Lemma 4.4. *For any arbitrary sequence R of n requests, we have that for all $t \in \{0, \dots, m-1\}$, $S_t = S'_t$, and that for all $t \geq m$, either $S_t = S'_t$ or $S'_t = S_t \cup \{0\} \setminus \{\min\{s \in S_t : s > 0\}\}$.*

Lower bounding the cost by the maximum gap δ_t. To bound $\mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m)]$, we analyze the gap $\delta_t := \min\{s \in S_t : s > 0\}$ between the unique available server in $S'_t \setminus S_t = \{0\}$ and the unique available server in $S_t \setminus S'_t = \{\min\{s \in S_t : s > 0\}\}$. If $S_t = S'_t$, then there is no gap and we define $\delta_t = 0$. The next lemma formally bounds $\mathbb{E}[\sum_{t=m+1}^{n}(cost_t(\mathcal{H}^{m-1}) - cost_t(\mathcal{H}^m)|\delta_m, S_m]$ as a function of the gap δ_t.

Lemma 4.5. *For all $m \in [n]$, we have that*

$$\mathbb{E}\Big[\sum_{t=m+1}^{n}(cost_t(\mathcal{H}^{m-1}) - cost_t(\mathcal{H}^m)|\delta_m, S_m\Big] \geq \frac{1}{2}\mathbb{E}\Big[\max_{t\in\{0,\dots,\min(t_{\{0\}},t_w)-m\}} \delta_{t+m} - \delta_m|\delta_m, S_m\Big] - \mathbb{P}(t^d > t_{\{0\}}|\delta_m, S_m),$$

where $s_{t,1} := \min\{s > 0 : s \in S_t\}$ and $s_{t,2} := \min\{s > s_{t,1} : s \in S_t\}$; $t_w := \min\{t \geq m : s_{t,2} - s_{t,1} > s_{t,1},$ or $s_{t,2} = \emptyset\}$, $t^d = \min\{t \geq m : \delta_t = 0\}$ and $t_{\{0\}} := \min\{t \geq m|\ S_t \cap \{0\} = \emptyset\}$.

To prove Lemma 4.5, we first show some structural properties of the process $\{(\delta_t, S_t)\}_{t\geq 0}$. In particular, we partially characterize the transitions from (δ_t, S_t) to (δ_{t+1}, S_{t+1}) (Lemma 4.9), and show that if at some time step t, there remains servers at 0 and the gap $s_{2,t} - s_{1,t}$ between the two first servers with positive location in S_t is smaller than the gap δ_t, then $\mathbb{E}[cost_t(\mathcal{H}^{m-1}) - cost_t(\mathcal{H}^m)]$ is lower bounded by $(\delta_t - \delta_{t-1})/2$.

Lower bounding the maximum gap δ_t. By Lemma 4.5, it remains to lower bound the maximum gap δ_t, for $t \geq m$. To analyze this gap, we first need to introduce some additional notation and terminology. We consider a partition $I_0, I_1, \dots$ of $(0, 1]$ into intervals of geometrically increasing size, where $I_i = (y_{i-1}, y_i]$ and $y_i = (3/2)^i n^{-1/5}$ (with the convention $y_{-1} = 0$). In addition, we say that a sequence of requests is *regular* if, for any $i \in [n]$, the number of requests between any time steps t and t' that are in the interval $[(i-1)/n, i/n]$ "sufficiently concentrates". More formally, we start by discretizing the interval $[0, 1]$ as $\mathcal{D} = \{\frac{i}{n} : i \in \{0, \dots, n\}\}$. For any interval $I = [i_L, i_R] \subseteq [0, 1]$, we also consider $d^+(I)$, the smallest interval with end points in $\mathcal{D}$ that contains I, and $d^-(I)$, the largest interval with end points in $\mathcal{D}$ contained in I.

(1) $d^+(I) := [d_L^+, d_R^+]$, with $d_L^+ := \max\{x \in \mathcal{D}|x \leq i_L\}$ and $d_R^+ := \min\{x \in \mathcal{D}|x \geq i_R\}$
(2) $d^-(I) := [d_L^-, d_R^-]$, with $d_L^- := \min\{x \in \mathcal{D}|x \geq i_L\}$ and $d_R^- := \max\{x \in \mathcal{D}|x \leq i_R\}$.

Definition 4.6. We say that a realization R of the sequence of requests is regular if for all $d, d' \in \mathcal{D}$ such that $d < d'$, and for all $t, t' \in [n]$ such that $t < t'$,

(1) $|\{j \in \{t, \dots, t'\}|\ r_j \in [d, d']\}| \geq (d' - d)(t' - t) - \log(n)^2\sqrt{(d' - d)(t' - t)}$,

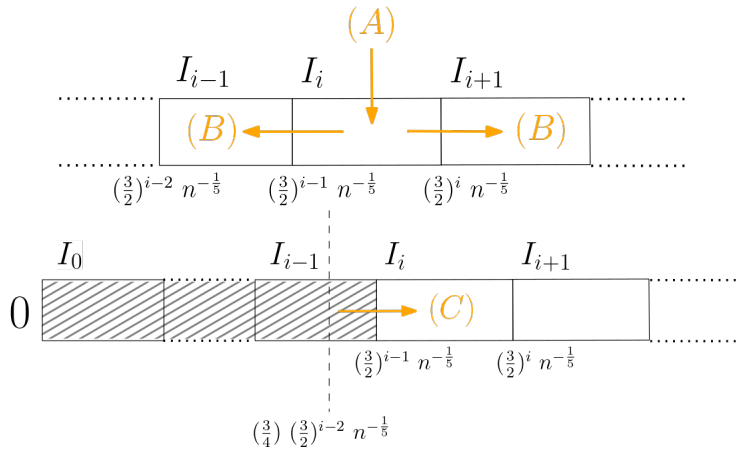

Fig. 4. Requests in and out of I_i up to time $\bar{t}_i := \min(t_i, t_{i-1} + c_2(n - t_{i-1}))$, with (A) the total number of requests that arrived in I_i from time 0 to $\bar{t}_i$, (B) the total number of requests that arrived in I_i and were matched outside I_i from time 0 to $\bar{t}_i$, and (C) the total number of requests that arrived in $[\frac{3}{4}y_{i-1}, y_{i-1}]$ and were matched inside I_i from time $t_{i-1} + 1 + c_1(n - t_{i-1})$ to time $\bar{t}_i$ (note that there are no free servers in the dashed area for times $t \geq t_{i-1}$).

(2) and if $(d' - d)(t' - t) = \Omega(1)$, then

$$|\{j \in \{t, \ldots, t'\}|\ r_j \in [d, d']\}| \leq (d' - d)(t' - t) + \log(n)^2\sqrt{(d' - d)(t' - t)}.$$

By standard concentration bounds, a sequence of requests is regular with high probability.

Lemma 4.7. *With probability at least* $1 - n^{-\Omega(\log(n))}$*, the sequence of requests is regular.*

Once the requests of sequence is assumed regular, all events that can be derived by successive applications of simple Chernoff bounds become deterministic events. In particular, when a sequence of requests is regular, we can bound, for algorithm $\mathcal{H}^m$, the gap $s_{t,j+1} - s_{t,j}$ between the j^{th} and $j + 1^{th}$ free servers $s_{t,j}$ and $s_{t,j+1}$ with positive location at time $t \in [(1 - o(1))n]$.

The main technical lemma of the proof of the $\Omega(\log(n))$-competitive ratio is to lower bound the maximum gap δ_t over all $t \geq m$, which we do in the next lemma, where c_1, d_1, c_3 are positive constants.

Lemma 4.8. *For all* $i \in [d_1 \log(n)]$ *and* $m \leq c_1 n$,

$$\mathbb{P}\Big(\max_{t \in \{m, \ldots, \min(n - n^{c_3}, t_{\{0\}})\}} \delta_t \geq y_{i-1} | R \text{ is regular}, \delta_m, S_m\Big) \geq \frac{\delta_m}{y_i} - n^{-\Omega(\log(n))}.$$

Challenges to prove Lemma 4.8. The main difficulty in proving Lemma 4.8 is that the value of δ_t at each time step t is dependent on the value of S_t. However, S_t lies in an exponentially-sized state space and it is difficult to compute the exact distribution of S_t at all time steps. The key idea is to separate the analysis of $(\delta_1, \ldots, \delta_n)$ and $(S_1, \ldots, S_n)$. We first show that with high probability, the servers in $(S_1, \ldots, S_n)$ become globally unavailable from left to right (see below an overview of the proof for a more precise statement). Then, we lower bound the probability that for any y and any arbitrary sequence of sets $(S_1 \supseteq \ldots \supseteq S_n)$, $\delta = 0$ before all servers in the interval $(0, y]$ have become unavailable. Combining these two properties leads to the desired result.

$r_{t+1} \in \ldots$	$[0, \frac{\delta_t}{2}]$	$[\frac{\delta_t}{2}, \frac{\delta_t+w_t}{2}]$	$[\frac{\delta_t+w_t}{2}, \delta_t + \frac{w_t}{2}]$	$[\delta_t + \frac{w_t}{2}, \delta_t + w_t]$	$[\delta_t + w_t, 1]$
S_{t+1}	$S_t \setminus \{0\}$	$S_t \setminus \{\delta_t\}$	$S_t \setminus \{\delta_t\}$	$S_t \setminus \{\delta_t + w_t\}$	$\exists s \in [\delta_t + w_t, 1] \cap S_t : S_t \setminus \{s\}$
δ_{t+1}	δ_t	0	$\delta_t + w_t$	δ_t	δ_t
$\mathbb{E}[\Delta\text{cost}_{t+1}\vert\ldots]$	≥ 0	≥ 0	$\geq \begin{cases} \frac{w_t}{2} & \text{if } w_t \leq \delta_t \\ 0 & \text{otherwise.} \end{cases}$	≥ 0	≥ 0

Table 1. Values of (δ_{t+1}, S_{t+1}) and expected value of Δcost_{t+1} conditioning on (δ_t, S_t) and on r_{t+1}, assuming that $S_t \cap \{0\} \neq \emptyset$, $\delta_t \neq 0$ and $|S_t \cap (\delta_t, 1]| \geq 1$, and where $w_t := s_{t,2} - s_{t,1}$.

Overview of the proof of Lemma 4.8. The proof consists of three main parts. The first one analyzes the sets of free servers $S_0 \supseteq \ldots \supseteq S_n$ obtained with algorithm $\mathcal{H}^m$ at each time step, the second one partially characterizes the values of (δ_t, S_t) and studies the first time $t \geq m$ such that $\delta_t = 0$. The last ones combines the first two parts.

Part 1 of the proof of Lemma 4.8. We say that an interval I is *depleted* at time t if $S_t \cap I = \emptyset$. We let $t_I := \min\{t \geq 0 | S_t \cap I = \emptyset\}$, i.e., t_I is the time at which I is depleted. For simplicity, we write t_i instead of t_{I_i}. We first show that (A) there exists a constant $c_2 \in (1/2, 1)$ such that if $t_{i-1} \leq n - (1-c_2)^{i-1}n$, then, $t_{i-1} < t_i$. Then, we show that (B) if $t_0 < \ldots < t_{i-1} \leq n - (1-c_2)^{i-1}n$ and $t_{i-1} < t_i$, then, $t_i \leq n - (1-c_2)^i n$. To show this last result, we lower bound the number of requests matched in I_i until time $\overline{t_i} = \min(t_i, t_{i-1} + c_2(n - t_{i-1}))$. We first show (see Figure 4) that

$$|\{j \in [\overline{t_i}] \ | s_{\mathcal{H}^m}(r_j) \in I_i\}| \geq \Big[|\{j \in [\overline{t_i}] : r_j \in I_i\}| - |\{j \in [\overline{t_i}] : r_j \in I_i, s_{\mathcal{H}^m}(r_j) \notin I_i\}|\Big] + |\{j \in \{t_{i-1} + 1 + c_1(n - t_{i-1}), \ldots, \overline{t_i}\} : r_j \in [\tfrac{3}{4}y_{i-1}, y_{i-1}], s_{\mathcal{H}^m}(r_j) \in I_i\}|.$$

We then lower bound each of these terms separately, using in particular the regularity of the requests sequence. We deduce from this lower bound that if $t_i > n - (1-c_2)^i n$, then the number of requests matched in I_i exceeds the initial number of free servers in I_i, which is a contradiction. Hence the bound $t_i \leq n - (1-c_2)^i n$. Finally, by combining properties (A) and (B), we show inductively that there is a constant $d_1 > 0$ such that the intervals $\{I_i\}_{i \in [d_1 \log(n)]}$ are depleted in increasing order, i.e. that $m < t_1 < \ldots < t_{d_1 \log(n)} \leq n - n^{c_3}$ and that $m < t_{\{0\}}$, which is the main result of this first part.

Part 2 of the proof of Lemma 4.8. We start by a partial characterization of the value of (δ_t, S_t) and of the difference of cost $\Delta\text{cost}_{t+1} := \text{cost}_{t+1}(\mathcal{H}^{m-1}) - \text{cost}_{t+1}(\mathcal{H}^m)$ between the costs incurred by $\mathcal{H}^{m-1}$ and $\mathcal{H}^m$ at time step t as a function of δ_t and S_t.

Lemma 4.9. *All following properties hold at any time $t \in \{m, \ldots, n-1\}$:*

(1) if $\delta_t = 0$, then for all $t' \geq t$, we have $\delta_{t'} = 0$ and $\Delta cost_{t+1} = 0$,
(2) if $S_t \cap \{0\} \neq \emptyset$, then $\Delta cost_{t+1} \geq 0$.
(3) if $S_t \cap \{0\} \neq \emptyset$, $\delta_t \neq 0$ and $|S_t \cap (\delta_t, 1]| \geq 1$, then the values of (δ_{t+1}, S_{t+1}) and the expected value of $\Delta cost_{t+1}$ conditioning on (δ_t, S_t) and on r_{t+1} are as given in Table 1, where $w_t := s_{t,2} - s_{t,1}$ and where we write $\mathbb{E}[\Delta cost_{t+1}|...]$ instead of $\mathbb{E}[\Delta cost_{t+1}|(\delta_t, S_t), S_t \cap \{0\} \neq \emptyset, \delta_t \neq 0, |S_t \cap (\delta_t, 1]| \geq 1, r_{t+1} \in \ldots]$.
(4) if $\delta_{t+1} \neq \delta_t$, then $S_{t+1} = S_t \setminus \{\delta_t\}$.
(5) $\mathbb{E}[\mathbf{1}_{S_t \cap \{0\} = \emptyset, \delta_t \neq 0} \cdot \Delta cost_{t+1} | (\delta_t, S_t)] \geq -\mathbf{1}_{S_t \cap \{0\} = \emptyset, \delta_t \neq 0} \cdot \mathbb{P}(\delta_{t+1} = 0 | (\delta_t, S_t))$.

We recall that for any interval $I \subseteq [0,1]$, $t_I := \min\{t \geq m |\ S_t \cap I = \emptyset\}$ is the time at which I is depleted, and that $t^d := \min\{t \geq m : \delta_t = 0\}$ is the time at which the gap disappears. Using the properties given in Lemma 4.9, we next show the following lemma.

Lemma 4.10. *Conditioning on the gap δ_m and available servers S_m, and for all $y \in [\delta_m, 1]$, we have*

$$\mathbb{P}\Big(\min(t_{(0,y]}, t_{\{0\}}) \leq \min(t^d, t_{\{0\}})\Big|\delta_m, S_m\Big) \geq \frac{\delta_m}{y}.$$

In other words, starting from a gap δ_m, the probability that the gap has not yet disappeared at the time all the servers in $(0, y]$ have been depleted, or that all the servers at location 0 are depleted before either of these events occurs, is lower bounded by $\frac{\delta_m}{y}$.

Part 3 of the proof of Lemma 4.8. Since we have shown in the first part that the intervals $\{I_j\}$ are depleted in increasing order of j, we have that just before the time t_{y_i} where $(0, y_i] = \cup_{j \leq i} I_j$ is depleted, none of the intervals I_j for $j < i$ have free servers left, hence $\min\{s > 0 : s \in S_{t_{y_i}-1}\} \in I_i$. Hence, if $\delta_{t_{y_i}-1} \neq 0$, we have by the definition of δ_t that $\delta_{t_{y_i}-1} = \min\{s > 0 : s \in S_{t_{y_i}-1}\} \in I_i = (y_{i-1}, y_i]$, which, in particular, implies $\delta_{t_{y_i}-1} \geq y_{i-1}$. Thus, to prove the desired result, it suffices to lower bound the probability that $\delta_{t_{y_i}-1} \neq 0$ and that $t_{y_i} \leq t_{\{0\}}$ and $t_{y_i} \leq n - n^{c_3}$. By using the second part, we show that it is lower bounded by $\frac{\delta_m}{y_i} - n^{-\Omega(\log(n))}$.

4.2.3 The main lower bound result. By combining the main lemma (Lemma 4.8) with Lemma 4.5, we can show the following bounds on $\mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m)]$.

Lemma 4.11.

(1) For any $m > c_1 n$, we have: $\mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) | r_m \in [0, y_0]] = -O(n^{-1/5})$.
(2) For any $m \leq c_1 n$, we have: $\mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) | r_m \in [0, y_0]] = \Omega(\log(n) n^{-1/5})$.
(3) For any $m \in [n]$, we have: $\mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) | r_m \in (y_0, 1]] = 0$.

The last lemma needed is the following bound on OPT.

Lemma 4.12. *For any $n \in \mathbb{N}$, the expected cost* OPT *of the optimal offline matching for our lower bound instance satisfies: $E[\mathtt{OPT}] = O(n^{3/5})$.*

By doing a telescoping sum over all $m \in [n]$ and using that $\mathcal{H}^n = \mathcal{A}$ and $\mathcal{H}^0 = \mathcal{G}$, we obtain from Lemma 4.11 and 4.12 the lower bound.

Theorem 1.4. *For online matching on the line in the random requests model, the greedy algorithm achieves an $\Omega(\log n)$-competitive ratio.*

Proof. Since $\mathcal{A}^0 = \mathcal{G}$ and $\mathcal{A}^n = \mathcal{A}$, we have that

$$\begin{aligned}
&\mathbb{E}[cost(\mathcal{G})] - \mathbb{E}[cost(\mathcal{A})] \\
&= \sum_{m=1}^{n} \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m)] \\
&= \sum_{m=1}^{n} \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) | r_m \in (y_0, 1]] \mathbb{P}(r_m \in (y_0, 1]) \\
&+ \sum_{m=1}^{c_1 n} \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^m) | r_m \in [0, y_0]] \mathbb{P}(r_m \in [0, y_0])
\end{aligned}$$

$$
\begin{aligned}
&+ \sum_{m=c_1 n+1}^{n} \mathbb{E}[cost(\mathcal{H}^{m-1}) - cost(\mathcal{H}^{m}) | r_m \in [0, y_0]] \mathbb{P}(r_m \in [0, y_0]) \\
&\geq 0 + \sum_{m=1}^{c_1 n} C' \log(n) n^{-1/5} n^{-1/5} - \sum_{m=c_1 n+1}^{n} C n^{-1/5} n^{-1/5} \qquad \text{(for some constants } C, C' > 0) \\
&= n^{3/5} \Big(C' (\log(n)(c_1 - \tfrac{1}{n}) - C(1 - c_1 - \tfrac{1}{n}) \Big) \\
&= \Omega(\log(n) n^{3/5}),
\end{aligned}
$$

where the inequality is by Lemma 4.11 and since $\mathbb{P}(r_m \in [0, y_0]) = \mathbb{P}(r_m \in [0, n^{-1/5}]) = n^{-1/5}$. Thus, $\mathbb{E}[cost(\mathcal{G})] \geq \mathbb{E}[cost(\mathcal{A})] + \Omega(\log(n) n^{3/5}) = \Omega(\log(n) n^{3/5})$. Since by Lemma 4.12 we have $\mathbb{E}[OPT] = O(n^{3/5})$, we conclude that $\frac{\mathbb{E}[cost(\mathcal{G})]}{\mathbb{E}[OPT]} = \Omega(\log(n))$. □

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation through the grant *CAREER: An algorithmic theory of matching markets*, by a Columbia Center of AI Technology (CAIT) in collaboration with Amazon faculty research award, and by a Columbia Center of AI Technology (CAIT) PhD Fellowship.

REFERENCES

Mohammad Akbarpour, Yeganeh Alimohammadi, Shengwu Li, and Amin Saberi. 2022. The Value of Excess Supply in Spatial Matching Markets. (2022), 62. https://doi.org/10.1145/3490486.3538375

Antonios Foivos Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. 2014. A o(n) -Competitive Deterministic Algorithm for Online Matching on a Line. In *Workshop on Approximation and Online Algorithms*.

Nick Arnosti. 2022. Greedy matching in bipartite random graphs. *Stochastic Systems* 12, 2 (2022), 133–150.

David Arthur, Bodo Manthey, and Heiko Röglin. 2009. K-means has polynomial smoothed complexity. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 405–414.

Eric Balkanski, Yuri Faenza, and Mathieu Kubik. 2022. The Simultaneous Semi-random Model for TSP. In *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 43–56.

Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Seffi Naor. 2007. An O(Log2k)-Competitive Algorithm for Metric Bipartite Matching. In *Proceedings of the 15th Annual European Conference on Algorithms* (Eilat, Israel) *(ESA'07)*. Springer-Verlag, Berlin, Heidelberg, 522–533.

Timothy Brown. 2016. Matchmaking in Lyft Line — Part 1. *Lyft Engineering* (2016). https://tinyurl.com/3sdrw7yc

Vaggos Chatziafratis, Tim Roughgarden, and Jan Vondrak. 2017. Stability and Recovery for Independence Systems. In *25th Annual European Symposium on Algorithms (ESA 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Michele Conforti and Gérard Cornuéjols. 1984. Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem. *Discrete applied mathematics* 7, 3 (1984), 251–274.

Bela Csaba and András Pluhár. 2007. A randomized algorithm for the on-line weighted bipartite matching problem. *Journal of Scheduling* 11 (07 2007). https://doi.org/10.1007/s10951-007-0037-5

Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. 2011. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *Proceedings of the 12th ACM conference on Electronic commerce*. 29–38.

Matthias Englert, Heiko Röglin, and Berthold Vöcking. 2014. Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. *Algorithmica* 68, 1 (2014), 190–264.

Matthias Englert, Heiko Röglin, and Berthold Vöcking. 2016. Smoothed analysis of the 2-opt algorithm for the general TSP. *ACM Transactions on Algorithms (TALG)* 13, 1 (2016), 1–15.

Uriel Feige. 2021. Introduction to Semirandom Models. *Beyond the Worst-Case Analysis of Algorithms* (2021), 189.

Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, and Cliff Stein. 2010. Online stochastic packing applied to display ad allocation. In *European Symposium on Algorithms*. Springer, 182–194.

Alan Frieze, Colin McDiarmid, and Bruce Reed. 1990. Greedy Matching on the Line. *SIAM J. Comput.* 19, 4 (1990), 666–672. https://doi.org/10.1137/0219045 arXiv:https://doi.org/10.1137/0219045

Bernhard Fuchs, Winfried Hochstättler, and Walter Kern. 2003. Online Matching On a Line. *Electronic Notes in Discrete Mathematics* 13 (03 2003), 49–51. https://doi.org/10.1016/S1571-0653(04)00436-6

Martin Gairing and Max Klimm. 2019. Greedy metric minimum online matchings with random arrivals. *Oper. Res. Lett.* 47, 2 (2019), 88–91. https://doi.org/10.1016/j.orl.2019.01.002

Gagan Goel and Aranyak Mehta. 2008. Online budgeted matching in random input models with applications to Adwords.. In *SODA*, Vol. 8. 982–991.

Anupam Gupta, Guru Guruganesh, Binghui Peng, and David Wajc. 2019. Stochastic Online Metric Matching. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 67:1–67:14. https://doi.org/10.4230/LIPIcs.ICALP.2019.67

Anupam Gupta and Kevin Lewi. 2012. The Online Metric Matching Problem for Doubling Metrics. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 7391)*, Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer (Eds.). Springer, 424–435. https://doi.org/10.1007/978-3-642-31594-7_36

Varun Gupta, Ravishankar Krishnaswamy, and Sai Sandeep. 2020. PERMUTATION Strikes Back: The Power of Recourse in Online Metric Matching. In *APPROX-RANDOM*.

Isobel Asher Hamilton. 2019. Uber drivers are reportedly colluding to trigger 'surge' prices because they say the company is not paying them enough. *Insider* (2019). https://www.businessinsider.com/uber-drivers-artificially-triggering-surge-prices-reports-abc7-2019-6

Nina Holden, Yuval Peres, and Alex Zhai. 2021. Gravitational allocation for uniform points on the sphere. *The Annals of Probability* 49, 1 (2021), 287 – 321. https://doi.org/10.1214/20-AOP1452

Joab Jackson. 2019. How Uber Eats Uses Machine Learning to Estimate Delivery Times? *The New Stack* (2019). https://thenewstack.io/how-uber-eats-uses-machine-learning-to-estimate-delivery-times/

Bala Kalyanasundaram and Kirk Pruhs. 1993. Online Weighted Matching. *J. Algorithms* 14, 3 (1993), 478–488. https://doi.org/10.1006/jagm.1993.1026

Bala Kalyanasundaram and Kirk Pruhs. 2000. The Online Transportation Problem. *SIAM J. Discret. Math.* 13, 3 (2000), 370–383. https://doi.org/10.1137/S0895480198342310

Yash Kanoria. 2021. Dynamic Spatial Matching. https://doi.org/10.48550/ARXIV.2105.07329

Yash Kanoria. 2022. Dynamic Spatial Matching. In *Proceedings of the 23rd ACM Conference on Economics and Computation* (Boulder, CO, USA) *(EC '22)*. Association for Computing Machinery, New York, NY, USA, 63–64. https://doi.org/10.1145/3490486.3538278

Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. 1990. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 352–358.

Elias Koutsoupias and Akash Nanavati. 2004. The Online Matching Problem on a Line. In *Approximation and Online Algorithms*, Roberto Solis-Oba and Klaus Jansen (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 179–191.

Marvin Künnemann and Bodo Manthey. 2015. Towards understanding the smoothed approximation ratio of the 2-opt heuristic. In *International Colloquium on Automata, Languages, and Programming*. Springer, 859–871.

Yiming Li, Jingzhi Fang, Yuxiang Zeng, Balz Maag, Yongxin Tong, and Lingyu Zhang. 2020. Two-sided online bipartite matching in spatial data: experiments and analysis. *GeoInformatica* 24, 1 (2020), 175–198.

Bodo Manthey and Heiko Röglin. 2013. Worst-case and smoothed analysis of k-means clustering with Bregman divergences. *Journal of Computational Geometry (Old Web Site)* 4, 1 (2013), 94–132.

Andrew Mastin and Patrick Jaillet. 2013. Greedy online bipartite matching on random graphs. *arXiv preprint arXiv:1307.2536* (2013).

Nicole Megow and Lukas Nölke. 2020. Online Minimum Cost Matching with Recourse on the Line. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference (LIPIcs, Vol. 176)*, Jaroslaw Byrka and Raghu Meka (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 37:1–37:16. https://doi.org/10.4230/LIPIcs.APPROX/RANDOM.2020.37

Aranyak Mehta. 2013. Online Matching and Ad Allocation. *Foundations and Trends in Theoretical Computer Science* 8 (4) (2013), 265–368. http://dx.doi.org/10.1561/0400000057

Aranyak Mehta et al. 2013. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science* 8, 4 (2013), 265–368.

Adam Meyerson, Akash Nanavati, and Laura Poplawski. 2006. Randomized Online Algorithms for Minimum Metric Bipartite Matching. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm* (Miami, Florida) *(SODA '06)*. Society for Industrial and Applied Mathematics, USA, 954–959.

Krati Nayyar and Sharath Raghvendra. 2017. An Input Sensitive Online Algorithm for the Metric Bipartite Matching Problem. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 505–515. https://doi.org/10.1109/FOCS.2017.53

Enoch Peserico and Michele Scquizzato. 2021. Matching on the line admits no $o(\sqrt{logn})$-competitive algorithm. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Sebastian Pokutta, Mohit Singh, and Alfredo Torrico. 2020. On the unreasonable effectiveness of the greedy algorithm: Greedy adapts to sharpness. In *International Conference on Machine Learning*. PMLR, 7772–7782.

Sharath Raghvendra. 2016. A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France (LIPIcs, Vol. 60)*, Klaus Jansen, Claire Mathieu, José D. P. Rolim, and Chris Umans (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:16. https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.18

Sharath Raghvendra. 2018. Optimal Analysis of an Online Algorithm for the Bipartite Matching Problem on a Line. In *34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Aviad Rubinstein and Junyao Zhao. 2022. Budget-Smoothed Analysis for Submodular Maximization. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Yongxin Tong, Jieying She, Bolin Ding, Lei Chen, Tianyu Wo, and Ke Xu. 2016. Online Minimum Matching in Real-Time Spatial Data: Experiments and Analysis. *Proc. VLDB Endow.* 9, 12 (2016), 1053–1064. https://doi.org/10.14778/2994509.2994523

Ying The Tsai, Chuan Yi Tang, and Yunn Yen Chen. 1994. Average Performance of a Greedy Algorithm for the On-Line Minimum Matching Problem on Euclidean Space. *Inf. Process. Lett.* 51, 6 (1994), 275–282. https://doi.org/10.1016/0020-0190(94)00116-2

Pan Xu, Yexuan Shi, Hao Cheng, John Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, Yongxin Tong, and Leonidas Tsepenekas. 2019. A unified approach to online matching with conflict-aware constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2221–2228.