


Online Facility Location with Weights and Congestion

Arghya Chakraborty ✉ 

Tata Institute of Fundamental Research, Mumbai, India

Rahul Vaze ✉

Tata Institute of Fundamental Research, Mumbai, India

Abstract

The classic online facility location problem deals with finding the optimal set of facilities in an online fashion when demand requests arrive one at a time and facilities need to be opened to service these requests. In this work, we study two variants of the online facility location problem; (1) weighted requests and (2) congestion. Both of these variants are motivated by their applications to real life scenarios and the previously known results on online facility location cannot be directly adapted to analyse them.

Weighted requests: In this variant, each demand request is a pair (x, w) where x is the standard location of the demand while w is the corresponding weight of the request. The cost of servicing request (x, w) at facility F is $w \cdot d(x, F)$. For this variant, given n requests, we present an online algorithm attaining a competitive ratio of $\mathcal{O}(\log n)$ in the secretarial model for the weighted requests and show that it is optimal.

Congestion: The congestion variant considers the case when there is a congestion cost that grows with the number of requests served by each facility. For this variant, when the congestion cost is a monomial, we show that there exists an algorithm attaining a constant competitive ratio. This constant is a function of the exponent of the monomial and the facility opening cost but independent of the number of requests.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, online facility location, probabilistic method, weighted-requests, congestion

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.6

Related Version *arXiv Version*: <https://arxiv.org/abs/2211.11961>

Funding Research supported by the Department of Atomic Energy, Government of India, under project 12-R&D-TFR-5.01-0500.

Arghya Chakraborty: Research supported in part by the Google India Research Award (PI: Prahladh Harsha).



Acknowledgements The first author wishes to extend his deepest thanks to Prof. Umang Bhaskar, Prof. Prahladh Harsha, and Prof. Jaikumar Radhakrishnan for the thought-provoking discussions and the valuable advice they generously shared with him on multiple occasions.

1 Introduction

The facility location problem is one of the most well-studied problems in the field of algorithms [14, 3, 17], where Meyerson [18] studied the online version of this problem where requests arrive in a sequence and on arrival, the request must be allocated to a facility irrevocably and a cost is incurred depending on the distance of the request from the allocated facility. New facilities can also be opened on the arrival of a new request so that the current request and future requests have a nearby facility to be allocated to. However opening facilities incur a cost too and the goal is to minimize the total cost incurred.



© Arghya Chakraborty and Rahul Vaze;

licensed under Creative Commons License CC-BY 4.0

43rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2023).

Editors: Patricia Bouyer and Srikanth Srinivasan; Article No. 6; pp. 6:1–6:22



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

To motivate the variants of online facility location we study, let us consider a plausible real-life scenario where such an online facility location algorithm may be deployed. Suppose the government is trying to set up vaccination centres in order to distribute vaccines depending on incoming requests. While online facility location can model the problem, a few other issues arise. For instance, one facility might not be able to satisfy a large number of requests and depending on the number of requests, there may be congestion. Also, there may be multiple vaccine doses resulting in weighted requests where each request needs to be satisfied a certain number of times (for instance, some people might require 2 doses while others require more¹). There are numerous other real life applications of the facility location problem. These additional variants involving congestion and weighted-requests have their own costs which need to be taken into account when designing a solution and the classical facility location problem may not be directly applicable in these cases. In this work, we generalize the classical facility location problem to incorporate these extra variants and prove matching upper and lower bounds. We study two variants of the classical facility location, each incorporating one of these variants. We now define the classical online facility location problem and these two variants formally.

1.1 Online Facility Location Problem

The facility location problem comprises of a set of requests $\{x_1, x_2, \dots, x_n\}$ on a metric space $(\mathcal{M}, d(\cdot, \cdot))$. The objective is to open a set of facilities F on this metric space. There is a facility opening cost of f incurred for opening each such facility and a distance cost for each request which is proportional to the distance of the request to its nearest facility. Naturally, the objective is to open facilities such that the total cost incurred is minimum, where the total cost incurred is given by

$$f \cdot |F| + \sum_{i=1}^n d(x_i, F)$$

where $d(x_i, F)$ denotes the distance of x_i to its nearest facility in F . This problem is known to be NP-Hard [12].

The online variant of the facility location problem was first studied in detail by Meyerson [18]. In this setting, a request, x_i , needs to be allotted *on its arrival* to a facility, F , either an existing facility or a newly-opened facility. The cost incurred on allotting the request is the distance cost $d(x_i, F^{(i)})$ where $F^{(i)}$ refers to the set of facilities available at the time of servicing request x_i (including possibly the newly-opened facility). As in the offline setting, opening new facilities costs f for every facility opened. Meyerson [18] constructed a randomized algorithm, *RFL* (Randomized Facility Location), which attains a competitive ratio of $\mathcal{O}(\log n)$ in expectation. Later, Fotakis [9] proved that no online algorithm can attain a competitive ratio better than $\Omega\left(\frac{\log n}{\log \log n}\right)$ and also did an improved analysis proving that *RFL* attains a competitive ratio of $\mathcal{O}\left(\frac{\log n}{\log \log n}\right)$ in expectation.

Meyerson also showed that in the secretarial model (where the requests are adversarial but the arrival order is uniformly at random), his algorithm attains a competitive ratio of 8.

¹ We assume that these multiple requests must be served by the same facility else these multiple requests may be treated as independent requests.

1.2 The two variants and our results

Online facility location problem has many applications in real world. However, as observed by the vaccination example, some of these applications may have additional constraints on the classical online facility problem. In this paper, we study two variants incorporating these additional constraints.

1.2.1 Variant I: Weighted Requests

In our first variant, we consider a setting of weighted requests, where each request x_i arrives with an additional weight w_i . One may view this weight w_i as the number of times the request x_i needs to be served or simply as a weight corresponding to a premium client.

Formally, a sequence of requests (x_i, w_i) arrive in an online fashion where x_i is a point in the metric space $(\mathcal{M}, d(\cdot, \cdot))$ while w_i is a positive real number. On arrival, the request (x_i, w_i) must be allocated to a facility, which could be one of the existing facilities or a newly-opened facility. As before, the facility opening cost is f but the cost incurred on allocating request (x_i, w_i) is $d(x_i, F^{(i)}) \cdot w_i$, instead of the usual $d(x_i, F^{(i)})$. If $w_i \cdot d(x_i, \cdot)$ is also a metric, this can be handled by the classical case, but this may not always be the case. As usual, the goal is to minimize the total cost incurred which is given by the following expression:

$$f \cdot |F| + \sum_{i=1}^n d(x_i, F) \cdot w_i.$$

Note, that while our motivating example suggested that the weights w_i 's are positive integers (corresponding to the number of times a request needs to be served), the above formulation is more general and allows for any positive real number.

We first modify Fotakis' lower bound to show that in the worst-case setting, no online algorithm can attain a competitive ratio better than $\Omega(n)$ matching the naïve algorithm that opens a new facility at every request. We then show that in the secretarial setting (when the requests are adversarial, but the ordering is random), there is an online algorithm that achieves a competitive ratio of $\mathcal{O}(\log n)$, which we show is tight once again by adapting Fotakis' lower bound. Observe that these results prove that this variant is provably different from the classical setting.

1.2.2 Variant II: Congestion

Our next variant incorporates the notion of congestion: if a facility is attending to multiple requests, there is an additional cost that it has to pay, depending on the number of requests it is attending to.

Formally, a sequence of requests x_i arrives and on arrival, the request must be allocated to a facility. Just as before, a facility may be opened in order to allocate x_i . A distance cost of $d(x_i, F^{(i)})$ is incurred and a facility opening cost of f is incurred every time a facility is opened. As of yet, this is exactly same as the online facility location problem. However we incorporate an additional congestion cost. If the facility to which x_i is allocated to has k requests allotted to it after x_i is allotted, then a congestion cost of $g(k) - g(k-1)$ is incurred in addition, where g is a convex non-decreasing function, satisfying $g(0) = 0$. We shall call g as the congestion function. In this model, if a total of ℓ requests are allocated to a facility, a total congestion cost of $g(\ell)$ is incurred and the goal is to minimize the total cost incurred. Note that the total cost includes three components: facility-opening costs, allocation costs and congestion costs.

In this work, we prove results for the special case when g is a monomial. We show that Meyerson's algorithm, RFL (Randomized Facility Location), can be modified to obtain a $\mathcal{O}(\log k^* / \log \log k^*)$ competitive ratio, where $k^* = 2 \cdot g^{-1}\left(\frac{f}{g(2)-2}\right)$. Note that this is independent of n , the number of requests. We also show that this is tight up to a constant by providing a matching lower bound.

A few words on the restriction of congestion function to monomials. Our results work for any convex non-decreasing congestion function g that satisfies $g(0) = 0$ and $g(a \cdot b) = g(a) \cdot g(b)$ for all a, b . Hence, the assumption that g is a monomial. While the assumption that the congestion function g is a monomial is restrictive, it is interesting that in this setting, we get tight matching upper and lower bounds. It would be interesting to extend this proof to more general convex functions.

1.2.3 Proof Techniques

While all the algorithms that we analyse are modifications of Meyerson's algorithm *RFL* and the lower bound analyses are inspired by Fotakis' analysis, significant modifications are needed to handle each variant.

For the weighted requests, as indicated earlier, we show that in the adversarial model, no algorithm can attain a competitive ratio better than $\Omega(n)$, which is trivially achievable by opening a facility for all the requests. Then we analyse this in the secretarial model, where we are able to connect this problem to a problem on increasing sequences in a uniformly random permutation. We obtain both the upper and lower bound of $\Theta(\log n)$ by studying properties of increasing sequences in random permutations. In particular, we reduce the analysis of the algorithm to the analysis of a process related to permutations, which we refer to as the *Selection Process*, which might be of independent interest. The Selection Process has the property that if the input permutation is chosen adversarially, its cost can be as high as $\Omega(n)$ while on a random permutation, the expected cost drops to $\Theta(\log n)$.

For the model with congestion, we first show that the optimal offline algorithm (*OPT*) will not allocate more than k^* requests to a facility. Using this fact, we analyze any arbitrary facility c^* opened by *OPT*. We analyze the cost incurred by our online algorithm over the requests allocated to c^* . We are able to split up this cost carefully into two components – one component is simply the congestion cost while the other component has no congestion cost involved. Using the fact that the facility c^* has at most k^* requests allotted to it, we are able to attain a competitive ratio of $\mathcal{O}\left(\frac{\log k^*}{\log \log k^*}\right)$.

1.3 Related Work

Since many real-life problems are similar to facility location problems, it has received a lot of attention. It has even been used as a tool to solve other online problems (like [13, 6]). The survey by Fotakis[10] gives a nice depiction of the problem and the general techniques that one needs to be familiar with. Other modifications that have been studied, include a setting where the facilities can themselves be moved with some cost [8], or the requests may be deleted [7]. The latter paper also studies capacitated facility location, which assumes that each facility can attend to at most a fixed number of requests. This is very similar to our congestion model, but as the bounds suggest, these two problems seem to be different.

Similarly, there has been interest on making these algorithms deterministic. Anagnostopoulos, Bent, Upfal, and Hentenryck in their paper [2] show that there is an $\mathcal{O}(\log n)$ -competitive deterministic algorithm for online facility location. Later Fotakis in [10] showed

a deterministic primal dual algorithm attaining $\mathcal{O}\left(\frac{\log n}{\log \log n}\right)$ competitive ratio. More recently Kaplan, Naori, and Raz in [16] show that when the requests arrive uniformly at random, no online algorithm can achieve a competitive ratio better than 2 and a modification of Meyerson's algorithm achieves a competitive ratio of 3 in expectation.

Another interesting topic in online algorithms is to improve the performance based on advice. This has been studied widely in [1, 4, 11, 15] and indeed one can have algorithms with better competitive ratio if the advice is good.

One thing to note is that in all of these papers, the algorithms in most cases are modifications of the original algorithm by Meyerson. The analysis is what makes these problems interesting. This is true for our results as well where the algorithms are adaptations of Meyerson's algorithm however the analysis needs some novel ideas.

2 Facility Location with Weighted Requests

In this model, we assume that input requests are ordered pairs of the form (x_i, w_i) where x_i describes the position of the request while w_i is the weight of the request. This could be the model for how long the request stays in the system or the number of times the request has to be serviced. The corresponding cost incurred is $w_i \cdot d(x_i, F)$ (as opposed to just $d(x_i, F)$). Our results hold for arbitrary positive weight units, which need not be integers.

► **Definition 2.1** (Weighted Online Facility Location). *A sequence of n ordered pairs (x_i, w_i) are given as input, where $x_i \in \mathcal{M}$ for some metric space \mathcal{M} . Each such request needs to be allocated to a facility in an online fashion. Opening a facility incurs a cost of f and allocating a request (x_i, w_i) to a facility F incurs a cost of $d(x_i, F) \cdot w_i$. The goal is to open facilities and allocate the requests in an online fashion such that the total cost incurred is minimized.*

Here, the algorithm knows the metric space \mathcal{M} but does not know n beforehand.

It is not hard to show that in the worst case setting, one can not expect to achieve a competitive ratio better than $\Omega(n)$ (See Section 4.2 for more details), which is trivially achievable by opening a facility on every request.

In light of the above, we study this setting in the secretarial model. Informally speaking, the secretarial model is one where the requests may be adversarial but the order in which they appear is uniformly at random. This ensures that the arrival order of the input requests cannot be adversarial. In other words, while the n input requests may be arbitrary, the secretarial model assumes that at each step all the remaining requests are equally likely to arrive as the next request. For the secretarial model, we show the following matching upper and lower bounds :

► **Theorem 2.2.** *In the online facility location problem with weighted requests, no online algorithm can obtain a competitive ratio better than $\Omega(\log n)$ in the secretarial model.*

► **Theorem 2.3.** *For the online facility location problem with weighted requests, there exists an algorithm attaining a competitive ratio of $\mathcal{O}(\log n)$ in expectation under the secretarial model.*

The proof of lower bound is deferred to Section 4 and the upper bound is proved in Section 5.

3 Facility Location with Congestion

In this model, we consider a sequence of n requests x_1, \dots, x_n from a metric space $(\mathcal{M}, d(\cdot, \cdot))$ endowed with a distance metric d . The goal is to open facilities when needed and assign these incoming requests to the facilities. We assume we are given a congestion function g , which is a convex non-decreasing function such that $g(0) = 0$. Opening a facility incurs a cost of f , while assigning a request x_i to a facility F containing k requests before x_i was assigned incurs a cost of $d(x_i, F) + g(k+1) - g(k)$. The additional $g(k+1) - g(k)$ is the congestion cost at the facility due to the new request.

► **Definition 3.1** (Online Facility Location with Congestion). *A sequence of n requests, x_i , is given as input, where $x_i \in \mathcal{M}$ for some metric space \mathcal{M} . Each such request needs to be allocated to a facility in an online fashion. Opening a facility incurs a cost of f and allocating a request x_i to a facility F containing k requests before x_i was assigned incurs a cost of $d(x_i, F) + g(k+1) - g(k)$. The goal is to open facilities and allocate the requests in an online fashion such that the cost incurred is minimized.*

In this paper, we consider congestion costs which satisfy $g(a \cdot b) = g(a) \cdot g(b)$ (for example a monomial). For this particular model, we prove the following :

- **Theorem 3.2.** *In the online facility location problem with congestion,*
1. *There exists an online algorithm attaining a competitive ratio of $\mathcal{O}(\frac{\log k^*}{\log \log k^*})$ in expectation, where $k^* := 2 \cdot g^{-1}(\frac{f}{g(2)-2})$, a constant independent of the number of requests.*
 2. *Furthermore, no randomized online algorithm can achieve a competitive ratio better than $\frac{\log k^*}{\log \log k^*}$.*

For want of space, the proofs of these theorems is deferred to Appendix B.

4 Lower bounds for the weighted-request variants

In this section we shall first show that in the worst case setting, no online algorithm can attain a competitive ratio better than $\Omega(n)$, which will be the motivation for us to consider the secretarial setting where we shall then show that no online algorithm may have a competitive ratio better than $\Omega(\log n)$ in expectation. This will make our analysis of Theorem 2.2 tight.

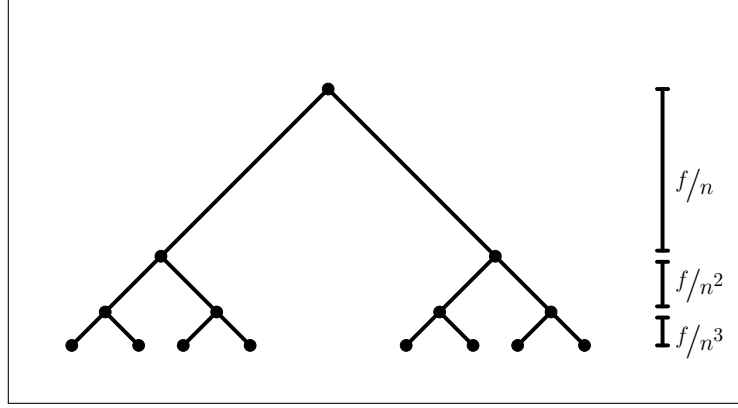
For the worst case setting, we shall work with a particular sequence of input requests. Then for the secretarial case, we shall use the same set of input requests assuming that the order of arrival of these requests is uniformly at random. Hence, we will first describe this set of requests.

4.1 The input for the lower bounds

The input requests will be constructed using a binary tree on a star metric. This is similar to the lower bound presented by Fotakis [9].

There will be requests along the nodes of the tree² in Figure 1, forming a path, starting from the root node and all the way to a leaf node. In order to refer to this tree we shall use the term “level”. The root node is at level 0, the children of the root node are level 1 and for any node, its level is 1 added to the level of its parent node, all the way to the leaves being level $n - 1$ nodes.

² The metric used for this analysis is the shortest path metric on the binary tree. However this can be embedded on the Euclidean Metric over \mathbb{R}^1 . This was shown in [9].



■ **Figure 1** Tree of possible requests.

The tree will be such that the distance between consecutive nodes will keep decreasing exponentially: the distance between a node at level i and a node at level $i + 1$ is $\frac{f}{n^{i+1}}$. We shall have 1 request at each node: the request at the i -th level node will have weight n^i .

For the worst case setting the requests will arrive in order of their levels - starting from the root node all the way to the leaf node. For the secretarial setting, the requests will be as mentioned but the arrival order will be uniformly at random.

4.2 Lower Bound in the Worst Case Setting

► **Proposition 4.1.** *No online algorithm can attain a competitive ratio better than $\Omega(n)$.*

Proof. For the mentioned input requests, an offline algorithm OFF may open 1 facility at the leaf node and in which case the distance of the root node to facility is at most $2\frac{f}{n}$ and for a node in the i -th level, its distance will be at most $2\frac{f}{n^{i+1}}$ from the leaf node. However taking into account the weight, the cost for the i -th level node is at most $2\frac{f}{n^{i+1}} \cdot n^i = 2\frac{f}{n}$. Hence the total cost paid by OFF is less than $f + \sum_{i=0}^{n-1} 2\frac{f}{n} = f + 2f = 3f$.

Let us now consider the performance of an online algorithm on this input. At every level the algorithm must either

- Open a new facility
- Or pay a distance cost (taking into account the weight of the request) to its parent node at the very least

This is because, when opening a new facility, the online algorithm may either open a facility on the node of the current request (say v_i) or try to guess future nodes and open a facility somewhere in the subtree of the current level node request. Both of these will result in a new facility nonetheless. Now if the algorithm tries to guess and open a future node in the subtree (let us say left sub tree of v_i) instead of opening a facility at the current node, the adversary will select the next node for input request from the other subtree (right subtree of v_i) thus adversarially ensuring that guessing never gives the algorithm a correct future node.

This ensures that the algorithm does not have any available facility in the subtree of the current request. Hence, if the algorithm decides to pay a distance cost, on an i -th level request, instead of opening a facility anywhere, it must pay a distance cost at least to its parent, $\frac{f}{n^i}$ with weight n^i resulting in a cost of f . The other case consists of opening a facility and in that case the algorithm incurs a cost of f anyway. Therefore the algorithm incurs a cost of f for each level of the tree, resulting in a total cost of nf .

One needs to also consider the case where the algorithm opens multiple facilities on a requested node ensuring that both the subtrees have facilities in them. However in this case the algorithm has already paid the facility opening cost multiple times. If the algorithm opens k facilities on a requested i -th level node (for $k \geq 2$), the algorithm cannot ensure that all the nodes in the $i + (k - 1)$ -th level below have facilities in their subtrees (since there are 2^{k-1} nodes at that level and $2^{k-1} \geq k$). This ensures that opening more than one facility on one node request is not beneficial for the algorithm.

Now let us compare the cost of the optimal offline algorithm, OPT , to the cost of an online algorithm. As we have already seen, the cost of any online algorithm is at least fn but the cost of OPT is at most $3f$ ensuring that $\frac{C_A}{C_{OPT}} \geq \frac{fn}{3f} = \frac{n}{3}$ for any online algorithm A .

Notice that this proof works against deterministic algorithms only. In order to prove this for randomized algorithms, we may use Yao's principle where as input we select one path among the 2^{n-1} paths uniformly at random. Alternatively, one can do this using the same technique (Lemma 4.5, in particular) used in Theorem 2.2, hence we have skipped the proof over here. ◀

A competitive ratio of n is trivially achievable by opening a facility on every request. However now, the naturally interesting question is whether an online algorithm can perform better if the input is secretarial.

4.3 Lower Bound in the Secretarial Setting

► **Definition 4.2.** *In the secretarial model, we assume that the adversary decides the input requests but not their arrival order. Instead after the adversary has decided the set of input requests, the arrival order of the requests is uniformly at random.*

Now, we shall give a proof for the lower bound in this setting.

Proof of Theorem 2.2. As mentioned earlier, we shall provide the same set of input requests but now the arrival order will be uniformly at random.

Notice that this makes the problem easier for an online algorithm in the sense that if an i -th level node is the first request to arrive, all the requests of level $0, 1, \dots, i - 1$ may be deduced by the online algorithm from the i -th level node and hence it will not have to open facilities for those requests when they arrive. Therefore we will not try to lower bound the cost of the algorithm on a request at level i if another request of higher level had already arrived earlier. However every time a node is requested such that no node below it has been requested yet, we shall ensure that the online algorithm has to pay some cost.

First we shall look at online algorithms, A , such that if it opens a facility on a request, it will open only at the location of the current request.

▷ **Claim 4.3.** Let A be an online algorithm such that on input requests, A would either open a facility at the request location or not open a facility at all. Then A cannot have a competitive ratio better than $\Omega(\log n)$.

Proof of Claim. As discussed earlier, the input will be composed in the following fashion :

- For an input of size n , we shall consider a binary tree on n levels : T_n .
- Then we shall choose a leaf node uniformly at random among the 2^{n-1} leaf nodes.
- There is a unique n length path connecting the root node to this leaf node.
- The input will comprise of requests on these nodes such that a request on the i -th level node will have weight of n^i .
- Also the distance between the $i - 1$ -th level node and the i -th level node will be $\frac{f}{n^i}$.

Notice that since we are in the secretarial model, these requested nodes (with corresponding weights) will arrive uniformly at random. Let us suppose that there are k requests v_1, v_2, \dots, v_k such that on arrival of the request v_i , it is the highest levelled node (or in other words, the lowest placed node in the tree structure) that has arrived till then.

Then for each of those requests, v_i , the online algorithm must either open a facility there or pay a distance cost with weight. Also, we have the guarantee that the nearest open facility available to the requested node is at least as far away as the distance to its parent node and hence when multiplied by the weight, the cost is exactly f , which is the same cost incurred for opening a facility. Therefore every time a request arrives such that no node of higher level has arrived yet, the online algorithm has to pay a cost of f . To finish off our proof we just need the following lemma:

► **Lemma 4.4.** *Let $S := [n]$ be the set containing the numbers 1 through n . Let π be a uniformly at random permutation of the set S . Let k_π be the number of times that the i -th element in the permutation π is the largest element observed till then. Then $\mathbb{E}_\pi[k_\pi] = \Theta(\log n)$.*

Proof. Let X_i be a random variable such that

$$X_i = \begin{cases} 1 & \text{if } \pi_i > \pi_j \text{ for all } j < i \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $\mathcal{P}[X_n = 1] = \frac{1}{n}$ since X_n will be 1 if and only if $\pi_n = n$ which has probability $\frac{1}{n}$. Also, conditioned on $\pi_{i+1}, \pi_{i+2}, \dots, \pi_n$, the probability that $X_i = 1$ is $\frac{1}{i}$ since no matter what π_{i+1}, \dots, π_n are, π_i will be the largest among $\pi_1, \pi_2, \dots, \pi_i$ with probability $\frac{1}{i}$. Hence $\mathbb{E}[k_\pi] = \mathbb{E}[\sum X_i] = \sum \mathbb{E}[X_i] = \sum \frac{1}{i} = \Theta(\log n)$. ◀

This Lemma 4.4 gives us that in the secretarial input model, in expectation there will be $\Theta(\log n)$ requested nodes such that they were the highest levelled nodes on their arrival. Hence cost of online algorithm A would be $\Omega(f \cdot \log n)$ compared to the offline optimal algorithm whose cost is at most $3f$ as we had calculated earlier (Here we are using the fact that the cost of the offline algorithm is the same for the secretarial input model as was for the non-secretarial model). Therefore the competitive ratio of online algorithm A is $\Omega(\log n)$. ◀

Now we consider the case that an online algorithm may not only open facilities at requested nodes but it may also open a facility at a nearby location instead of opening the facility exactly on the request. This potentially allows the online algorithm to guess future nodes and maybe reduce the cost. We shall look at the same input that we worked with, in the Claim 4.3, and observe that such a guessing algorithm cannot do much better in expectation.

► **Lemma 4.5.** *Let T_n be a binary tree with a path chosen uniformly at random from the root node to a leaf node. If a node, v , of this path is revealed, and an algorithm A opens a facility in the subtree of v , the guess will match with the actual path on 2 nodes (apart from v) in expectation.*

Proof. Since the path was chosen uniformly at random, A can guess the next level node with probability $1/2$. Also, 2 nodes can be guessed correctly by A with probability $1/4$ and continuing in this fashion, the probability that A 's guess matches with the path on i nodes is at most $\frac{1}{2^i}$.

6:10 Online Facility Location with Weights and Congestion

Therefore the expected number of nodes that A can guess correctly by opening a facility is at most

$$S := \sum_{i=1}^{\infty} \frac{i}{2^i}.$$

Firstly, we can use Ratio Test to observe that this series is actually converging. Then we notice that

$$2S = \sum_{i=1}^{\infty} \frac{i}{2^{i-1}} = \sum_{i=0}^{\infty} \frac{i+1}{2^i} = 1 + \sum_{i=1}^{\infty} \frac{i+1}{2^i} = 1 + \sum_{i=1}^{\infty} \frac{i}{2^i} + \sum_{i=1}^{\infty} \frac{1}{2^i} = 1 + S + 1$$

$$\implies S = 2.$$

◀

This means that every time the algorithm A receives a node which is the highest level till then, it has the choice to pay a distance cost with weight, or open a facility at the request itself or opening a facility in the subtree of the node, guessing future nodes. We have already seen the competitive ratio for algorithms that do not guess future nodes. Let k be the number of nodes observed such that they were the highest levelled nodes seen on their arrival. Then we had seen that an algorithm that does not guess incurs a cost of f each time resulting in a cost of $k \cdot f$ and then we had observed that the expected value of k is of the order of $\log n$ to obtain the final competitive ratio.

Now however the algorithm may guess the future nodes and potentially incur a less cost. Let k be the number of nodes observed such that they were the highest levelled nodes seen on their arrival. Then we can show that A must incur a cost of $f \cdot \frac{k}{3}$ at least, in expectation. If A incurs a cost less than $f \cdot \frac{k}{3}$, then algorithm A has opened $< \frac{k}{3}$ facilities, and hence has $< \frac{k}{3}$ guesses. Therefore the total number of nodes that A has guessed correctly is $< 2 \cdot \frac{k}{3}$ in expectation (Using Lemma 4.5). Note that on the arrival of node, say v , even if the algorithm A might guess a node, say v_1 , correctly (Where v_1 lies in the subtree of v), it may not reduce the cost incurred by A . This is because the next highest levelled node to arrive might be a child of v_1 in which case A needs to pay a cost of f anyway. In other words, correctly guessing a node might not reduce the number of highest levelled nodes observed, because the nodes guessed correctly might not be the highest levelled nodes to arrive. Therefore, taking into consideration the $< \frac{k}{3}$ nodes that A has paid for and the nodes that A has potentially guessed correctly, A has satisfied $< \frac{k}{3} + \frac{2k}{3} = k$ nodes in expectation (even after considering facilities opened, correctly guessed nodes and distance costs paid). This cannot be the case since A has to satisfy all the k nodes which were the highest levelled nodes on arrival.

Therefore A must incur a cost of $f \cdot \frac{k}{3}$ at least, in expectation. Since $k = \Theta(\log n)$ in expectation (Using Lemma 4.4), A must incur a cost of $\Omega(\log n)$ in expectation. ◀

5 Upper Bound for Weighted-Requests in the Secretarial Setting

Now we shall focus our attention on the setting with weighted requests and present the proof of Theorem 2.3.

In order to show this we will reduce the problem to a completely different problem, which we shall call *The Selection Process*. This is a novel idea and connecting the Weighted Facility Location Problem to the Selection Process is what allows us to get a tight upper bound of $\mathcal{O}(\log n)$.

► **Definition 5.1** (The Selection Process). Let $S = [n]$. A sequence of (p_i, j_i) arrive in an online fashion where $p_i \in [0, 1]^n$ is a vector of probabilities and $j_i \in S$ for $i \in [n]$. The first pair to arrive is (p_1, j_1) and we shall **select** the element j_1 with probability p_{1,j_1} .

Similarly, on input (p_i, j_i) , we shall select the element j_i with probability p_{i,j_i} but only if the element j_i is greater than the previously selected numbers.

Also, for all $1 \leq i \leq n-1$, and $j_i \in S$, we shall assume that $p_{i,j_i} \geq p_{i+1,j_i}$. The final quantity that we want to compute is the expected number of selected elements.

Let us first consider the case when the adversary selects both the p_i 's and the j_i 's. In this setting one can see that in the worst case, there can be as many as n elements selected. For this, consider the probability vectors p_i 's to be all 1's throughout and the element arrival order be $j_1 = 1, j_2 = 2, \dots, j_n = n$. In this case, all the elements will be selected.

Hence, let us consider the case when the adversary decides the probability vectors p_i 's (satisfying the constraint of $p_{i,j_i} \geq p_{i+1,j_i}$) however the arrival order of the requests is uniformly at random. That is, we take a permutation, π of $[n]$ uniformly at random and at step i , the element appearing is π_i . In this secretarial setting, one can show that no matter what probabilities the adversary chooses, the number of selected elements is $\mathcal{O}(\log n)$ in expectation.

► **Claim 5.2.** The number of selected elements in The Selection Process is $\mathcal{O}(\log n)$ in expectation.

We shall prove Claim 5.2 later.

► **Remark 5.3.** Notice that the selection process does not necessarily choose the longest increasing subsequence since the longest increasing subsequence in a uniformly permuted array, is of the size of $\Theta(\sqrt{n})$. It is actually enough to observe that this is $\Omega(n)$, which follows from Erdős-Szekeres theorem [19].

On the other hand, Lemma 4.4 tells us that greedily selecting the largest element gives us an increasing subsequence of size $\Theta(\log n)$.

However the adversary may manipulate the probability vectors p_i 's in a manner to make the largest increasing subsequences to be more likely to be picked. Indeed, in Remark 5.4, we show that if the probability vectors aren't enforced to be non-increasing, the adversary can make sure that there are $\Theta(\sqrt{n})$ selections in expectation. This is why the Claim 5.2 is of utmost importance.

► **Remark 5.4.** The fact that for all $j \in S$, and $1 \leq i \leq n-1$, we require $p_{i,j} \geq p_{i+1,j}$ is necessary. One may look at the example where the first \sqrt{n} probability vectors are such that they have 1 in their first \sqrt{n} terms and 0 everywhere else. The next \sqrt{n} probability vectors have their first \sqrt{n} terms 0, next \sqrt{n} terms as 1 and all 0's after that. This continues for \sqrt{n} many blocks of probability vectors, each with their corresponding block of 1's and remaining 0's. It is easy to see that given these probability vectors and a uniform arrival order of elements, one can expect $\Theta(\sqrt{n})$ elements to be selected in The Selection Process and $\mathcal{O}(\log n)$ would have been unachievable.

First, let us present our algorithm for facility location with weighted requests. Then we shall analyse the competitive ratio and see where The Selection Process comes in.

Algorithm 1 Weighted Randomized Facility Location (WRFL).

```

1: procedure  $WRFL(x_i, w_i)$  ▷ Input request is  $x_i$  with weight  $w_i$ 
2:    $F_i \leftarrow \text{Nearest facility to } x_i$ 
3:    $p_i \leftarrow \min\{1, \frac{d(x_i, F_i) \times w_i}{f}\}$ 
4:   With probability  $p_i$  open a new facility at  $x_i$  and allocate  $x_i$  to it
5:   With probability  $1 - p_i$ , assign  $x_i$  to  $F_i$ 
6: end procedure

```

While algorithm $WRFL$ is a simple modification of Algorithm 2 (Meyerson's Algorithm RFL , which we describe later), the analysis is completely different. The analysis of RFL does not go through because of the weight component. This is also apparent from the fact that the competitive ratios are of different order : $\log n$ for $WRFL$ while $\frac{\log n}{\log \log n}$ for RFL . Hence the entire structure of the analysis is different.

Now, we shall prove Theorem 2.3 for the Algorithm 1 in particular.

Proof of Theorem 2.3. In order to analyse the algorithm, we shall divide the run of the algorithm into what we shall define as phases. However before that, we shall set up some notations and observe a few properties.

Let c^* be a facility opened by OPT and let S_{c^*} be the set of requests that OPT assigns to c^* . Also when we say that an algorithm incurs a cost of C over a subset of requests, say T , we will look at each request in the subset T and note how much the algorithm paid for that request when allocating it- be it distance cost or facility opening cost. For example, if the algorithm allocates the request to a facility which was already opened on a previous request, not from T , then we just take into account the distance cost without considering the facility opening cost for that facility.

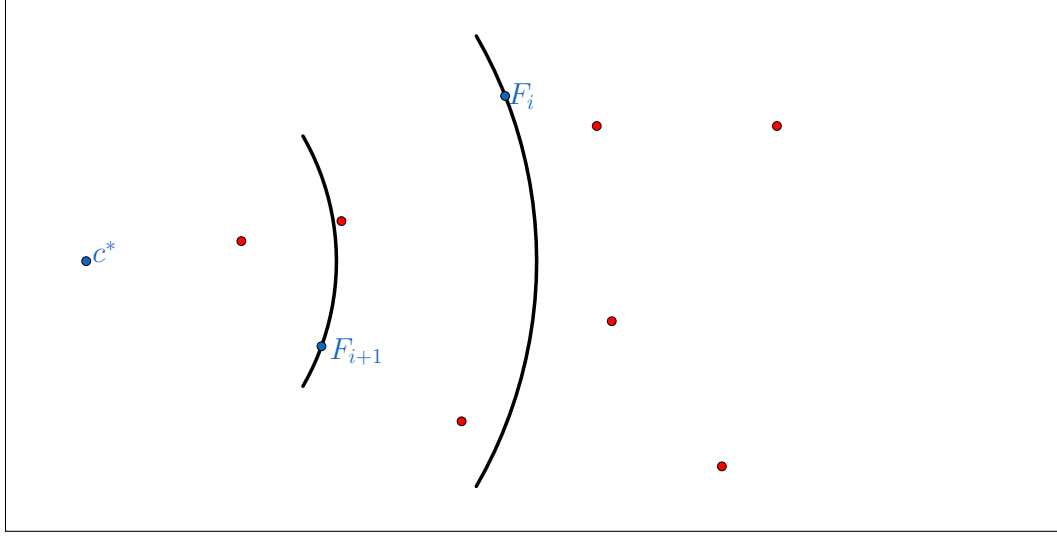
We may focus our attention on one facility c^* , opened by OPT and look at the set of requests S_{c^*} that OPT assigns to c^* . It is easy to see that analysing one such cluster is sufficient for our analysis (an explanation is provided in the extended paper [5]). For the sake of convenience, we shall refer to this set of requests as S , whenever we mean S_{c^*} .

Now, in order to analyze this, we shall define the phases of the algorithm.

► **Definition 5.5.** *Corresponding to the facility c^* , opened by OPT , the run of the algorithm $WRFL$ can be divided into phases as follows :*

- *The algorithm is said to be in Phase 0 until $WRFL$ opens a facility, say F_1 , on a request in S , at which point phase 1 starts.*
- *Phase 1 starts when $WRFL$ opens its first facility on a request of S (which we called F_1) and continues until $WRFL$ opens another facility, say F_2 on a request of S and this facility is closer to c^* than F_1 .*
- ⋮
- *Phase i starts when $WRFL$ opens a facility F_i and continues until $WRFL$ opens another facility, say F_{i+1} , on a request of S and this facility, F_{i+1} , is closer to c^* than F_i .*

Observe that this definition of Phase is valid for the given cluster of requests S . With respect to another cluster, the phases would be different. Also, the algorithm does not know about the phases, it is only well defined during the analysis of the algorithm. Just to get a clear idea, we can note that the number of phases depends not only on which requests $WRFL$ opens a facility on, but it also depends on the arrival order of the requests.



■ **Figure 2** Phase i starts when Facility F_i is opened and ends when Facility F_{i+1} is opened where by definition F_{i+1} should be closer to c^* than F_i . All the requests in Phase i are shown in red dots. They may be closer or farther away from F_i with respect to c^* . The only guarantee that we have is that the first request closer than F_i that results in a facility opening would change the phase to $i+1$.

► **Lemma 5.6.** *Consider a set of requests, S , assigned to the same facility by OPT . Assume that the request arrivals and $WRFL$'s random choices are such that there are k phases. Then the cost ratio, for the requests in S , of $WRFL$ to OPT is $\mathcal{O}(k)$ in expectation.*

Proof. Let us first compute the cost of OPT over the requests in S . We note first that OPT has only opened one facility for these requests, at c^* . This constitutes a cost of f . Other than this, OPT incurs a cost of $C := \sum_{x_i \in S} d(x_i, c^*) \cdot w_i$.

To complete the proof, we shall use two properties in order to analyse the cost of $WRFL$.

► **Property 5.7.** *Let $X = \{(x_1, w_1), (x_2, w_2), \dots, (x_m, w_m)\}$ be a subsequence of m requests. Then $WRFL$, in expectation, pays a cost $\leq f$ before it opens the first new facility in X .*

(This holds even if there are several open facilities at the beginning and even if new facilities are opened between two requests in the subsequence)

► **Property 5.8.** *On input request (x_i, w_i) , let F be the nearest facility to x_i . Then the cost that $WRFL$ pays in expectation for the request (x_i, w_i) is at most $2d(x_i, F) \cdot w_i$.*

The proofs of these two Properties have been studied in earlier works (e.g. [9, 10]), but for the sake of completion we have added them in Appendix C.

Using Property 5.7, algorithm $WRFL$ would incur a cost of f in expectation before it opens a facility on a request of S . This would result in phase changing from Phase 0 to Phase 1. Now let us say that we are at Phase i and we want to estimate $WRFL$'s cost over the requests in Phase i , where Phase i started when $WRFL$ opened a facility F_i . Here F_i is closer than F_{i-1} to c^* , by construction.

Type 1 requests. Let S_1 be the set of requests such that $d(c^*, x) < d(c^*, F_i)$ for all $x \in S_1$. Then using Property 5.7, algorithm $WRFL$ pays a cost of f in expectation over requests in S_1 before it opens a facility F_{i+1} on a request with $d(c^*, F_{i+1}) < d(c^*, F_i)$. This event of opening a facility ends up changing the phase from i to $i+1$. Hence for phase i , requests closer to c^* than F_i incur a cost of at most f in expectation.

Type 2 requests. Let S_2 be the set of requests such that $d(c^*, x) \geq d(c^*, F_i)$ for all $x \in S_2$. Now observe that OPT incurs a cost of $d(c^*, x) \cdot w_x$ for the request x (Where w_x denotes the weight of the request x). Let us now try to estimate the cost incurred by $WRFL$ for this request x . Here, $WRFL$ has an open facility F_i and $d(F_i, x) \leq d(x, c^*) + d(c^*, F_i)$ using triangle inequality. Also, $d(c^*, x) \geq d(c^*, F_i)$ by assumption. So $d(F_i, x) \leq d(x, c^*) + d(x, c^*) = 2d(x, c^*)$. Therefore the nearest facility to x for $WRFL$ is at most at a distance of $2d(x, c^*)$. Hence by Property 5.8, the cost incurred by $WRFL$ for the request x is at most $4d(x, c^*) \times w_x$ in expectation, which is 4 times the cost incurred by OPT itself.

Hence the total cost of Type 2 requests for $WRFL$ is at most $4C$ in expectation (Where OPT had incurred a cost of $f + C$ for the requests in S). On the other hand, $WRFL$ incurs a cost of f for Type 1 requests in expectation, for each phase. Also $WRFL$ incurs a facility opening cost of f for each phase. Therefore the total cost incurred by $WRFL$ is at most $2k \cdot f + 4C$ in expectation. This results in an expected competitive ratio of $2k + 4 = \mathcal{O}(k)$. ◀

► **Lemma 5.9.** *The expected number of phases is $\mathcal{O}(\log n)$.*

Proof. In order to view this we will reduce the problem to The Selection Process (Definition 5.1), which we had mentioned earlier.

We shall also assume the Claim 5.2 for now and finish the proof of Lemma 5.9, after which we shall give a formal proof of Claim 5.2. Let the requests in S be ordered in decreasing distance from c^* and labelled accordingly. Thus x_1 is the request farthest from c^* , x_2 is the second farthest request and so on till x_n which is the request nearest to c^* .

Just before the first request from S arrives online, there may be open facilities due to other requests. Given the configuration of the facilities (opened by $WRFL$) at this stage, each request $x_i \in S$ has a certain probability of opening a facility if it were the first request to arrive in S . This vector of probabilities will be our p_1 with $p_{1,i}$ being the probability for x_i opening a facility, if it was the first request to arrive. Now one of the requests x_{π_1} , arrives uniformly at random. The index of the request arriving corresponds to the element arriving in The Selection Process. Note that the vector of probabilities thus generated, depends on the previous requests and whether these requests have opened new facilities or not. However, this still adheres to the Selection Process, as long as the requests arrive uniformly at random.

At any stage when the i -th request x_{π_i} arrives, we can similarly compute the vector p_i and also this request would change the phase only if a facility is opened there and also if it is nearer to c^* than all the other opened facilities, which corresponds to the index of the request being larger than the indices of all the opened facilities. So, any time the phase changes, a facility is opened which is nearer to c^* than all previously opened facilities. This corresponds to the element being selected in the Selection Process.

Also while these probability vectors may be arbitrary, the probability of opening a facility on a particular request can only decrease over time. This is because if the request arrives later on, it can potentially have nearby facilities, reducing the probability of opening a facility but this probability of opening a facility cannot increase with time, implying $p_{i,j} \geq p_{i+1,j}$.

Therefore, if we can show that the number of selected elements in The Selection Process is $\mathcal{O}(\log n)$ in expectation, then we would have shown that the number of phases is $\mathcal{O}(\log n)$ in expectation, as was required. ◀

Using Lemmas 5.6 and 5.9, it follows that the competitive ratio of $WRFL$ is $\mathcal{O}(\log n)$. ◀

Proof of Claim 5.2. In order to prove this, we shall reduce the problem one more time.

Notice that the i -th element to arrive is π_i and is selected with probability p_{i,π_i} , if no larger element has yet been selected. We can view the selection probability of π_i as

$$p_{i,\pi_i} = p_{1,\pi_i} \times \frac{p_{2,\pi_i}}{p_{1,\pi_i}} \times \frac{p_{3,\pi_i}}{p_{2,\pi_i}} \times \dots \times \frac{p_{i,\pi_i}}{p_{i-1,\pi_i}}.$$

Since $p_{j,\ell} \geq p_{j+1,\ell}$ for all j (by definition), each of the terms $\frac{p_{j+1,\pi_i}}{p_{j,\pi_i}}$ is less than or equal to 1. Hence we can view the terms, $\frac{p_{j+1,\pi_i}}{p_{j,\pi_i}}$, as probabilities.

Now just before the first element π_1 arrives, we can toss random coins for each i such that the i -th coin is 1 with probability $p_{1,i}$. Given that π_1 is the first element to arrive, we can look at the realization of p_{1,π_1} and select it if it is 1.

Now just before the second element arrives, we can toss random coins for each i such that the i -th coin is 1 with probability $\frac{p_{2,i}}{p_{1,i}}$. Now, after observing π_2 , the second element to arrive, we can look at the realizations of p_{1,π_2} and $\frac{p_{2,\pi_2}}{p_{1,\pi_2}}$ and select the element π_2 only if both the entries are 1, and additionally if $\pi_2 > \pi_1$. Notice that π_2 is selected with probability p_{2,π_2} in this manner, as needed.

Similarly this process continues for the i -th element arriving, π_i , for all $i \geq 3$. Here we toss random coins for each element i that are 1 with probability $\frac{p_{i,\pi_i}}{p_{i-1,\pi_i}}$ and we take into account the realizations of the previous coin tosses of probability $\frac{p_{i',\pi_i}}{p_{i'-1,\pi_i}}$ (for all $i' < i$). The element π_i is selected only if all of the realizations are 1 and π_i is larger than all selected elements. One can see that selecting the elements in this manner produces the same probability of an element being selected.

However now we can state that an element π_i that has not yet arrived will certainly not be selected on its arrival if even one of the realizations $\frac{p_{i',\pi_i}}{p_{i'-1,\pi_i}}$ or p_{1,π_i} happens to be 0 (where $i' < i$, since π_i has not yet arrived), also we can say that the element π_i will certainly not be selected on its arrival if an element larger than π_i has already been selected.

Let us take one such array $A[.][.]$ of realizations of the probabilities. Where $A[i][j]$ is 1 with probability $\frac{p_{i,j}}{p_{i-1,j}}$. On the arrival of an element, π_i , we look at the π_i -th column and select the element π_i only if all the first i entries of the column are 1. Hence once a 0 appears in a column, that element will certainly not be selected and instead of tossing further coins for the element, we shall set all the remaining entries in the column to be 0. This maneuver does not alter the probability of any arriving element being selected. Also if a larger element had already been selected, the corresponding element will certainly not be selected.

▷ **Claim 5.10.** Let i be the first index when selection happens and π_i be the first element to be selected. Then either $\mathbb{E}[\pi_i] \geq \frac{n}{3}$ or the i -th row of A has at least $\frac{n}{3}$ entries as 0.

Proof. Let the i -th row of A have at least $\frac{2n}{3}$ entries as 1. This means that the columns corresponding to those $\frac{2n}{3}$ entries have 1 throughout, for the first i entries. We shall then conclude that $\mathbb{E}[\pi_i] \geq \frac{n}{3}$ to complete this proof.

Let c_1, c_2, \dots, c_k be the columns of A with 1's throughout till the i -th row. By our assumption, $k \geq \frac{2n}{3}$. Under the condition that the first element to be selected is the i -th element, the expected value of the selected element equals

$$\mathbb{E}[\pi_i] \stackrel{(1)}{=} \frac{1}{k} \sum_{j=1}^k c_j \stackrel{(2)}{\geq} \frac{1}{k} \sum_{j=1}^k j = \frac{k+1}{2} \geq \frac{\frac{2n}{3}+1}{2} \geq \frac{n}{3}.$$

		Element $j \rightarrow$							
Outcome of coin toss of i th iteration \rightarrow	p_1	1	1	1	1	0	1	1	1
	$\frac{p_2}{p_1}$	0	1	1	1		1	0	1
	$\frac{p_3}{p_2}$		1	0	1		0		1
	$\frac{p_4}{p_3}$		1		0				1
	$\frac{p_5}{p_4}$		1						0
			0		$\frac{p_{i,j}}{p_{i-1,j}}$				
	$\frac{p_n}{p_{n-1}}$								

■ **Figure 3** Array A of probabilities.

Equality (1) follows from the fact that because of the secretarial model, all the elements c_j are equally likely to be the i -th element. Since the i -th element, π_i , is the first selected element, π_i has to be one of the elements $\{c_1, c_2, \dots, c_k\}$ only. Also none of the elements c_j have appeared earlier, else they would have been selected first. Hence all of those k elements are equally likely to be π_i .

Inequality (2) holds because all the k elements are distinct integers from $[n]$. Therefore the sum will be the smallest when the k elements are $1, 2, \dots, k$.

Hence we have that if the i -th row of A has at least $\frac{2n}{3}$ entries as 1 then $\mathbb{E}[\pi_i] \geq \frac{n}{3}$, which proves our claim. \triangleleft

Applying Claim 5.10 recursively completes the proof. When the first element is selected, in expectation there will be $\frac{n}{3}$ elements that cannot be selected - either because they are smaller than the element picked or because they already have 0's in their columns in A . Therefore we can *discard* all those elements that can no longer be selected - meaning, if element j can no longer be selected we may remove the j -th row and the j -th column from the matrix A . Now we will have a smaller matrix, $A_1[.][.]$, leaving apart the rows and columns of the discarded elements. On this matrix, A_1 , we shall again apply the Claim 5.10 to discard $\frac{1}{3}$ fraction of its elements in expectation, when the next element is selected. This process continues and Claim 5.10 states that each time an element is selected, $\frac{1}{3}$ rd of the remaining elements are discarded in expectation. Hence this process ends in $\mathcal{O}(\log n)$ many steps in expectation. In other words, the number of elements selected is $\mathcal{O}(\log n)$. \triangleleft

References

- 1 Matteo Almanza, Flavio Chierichetti, Silvio Lattanzi, Alessandro Panconesi, and Giuseppe Re. Online facility location with multiple advice. In *Advances in Neural Information Processing Systems*, volume 34, pages 4661–4673. Curran Associates, Inc., 2021. URL: <https://proceedings.neurips.cc/paper/2021/file/250473494b245120a7eaf8b2e6b1f17c-Paper.pdf>.
- 2 Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Inf. Comput.*, 194(2):175–202, 2004. doi:10.1016/j.ic.2004.06.002.

- 3 Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 106–113, New York, NY, USA, 1998. Association for Computing Machinery. doi:10.1145/276698.276718.
- 4 Yossi Azar, Debmalaya Panigrahi, and Noam Touitou. Online graph algorithms with predictions. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 35–66, 2022. doi:10.1137/1.9781611977073.3.
- 5 Arghya Chakraborty and Rahul Vaze. Online facility location with timed-requests and congestion, 2022. arXiv:2211.11961.
- 6 Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 30–39, New York, NY, USA, 2003. Association for Computing Machinery. doi:10.1145/780542.780548.
- 7 Marek Cygan, Artur Czumaj, Marcin Mucha, and Piotr Sankowski. Online Facility Location with Deletions. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2018.21.
- 8 Björn Feldkord and Friedhelm Meyer auf der Heide. Online facility location with mobile facilities. In Christian Scheideler and Jeremy T. Fineman, editors, *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, Vienna, Austria, July 16-18, 2018*, pages 373–381. ACM, 2018. doi:10.1145/3210377.3210389.
- 9 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, January 2008. doi:10.1007/s00453-007-9049-y.
- 10 Dimitris Fotakis. Online and incremental algorithms for facility location. *SIGACT News*, 42(1):97–131, 2011. doi:10.1145/1959045.1959065.
- 11 Dimitris Fotakis, Evangelia Gergatsouli, Themis Gouleakis, and Nikolas Patrís. Learning augmented online facility location. (manuscript), 2021. arXiv:2107.08277.
- 12 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12(3):133–137, 1981. doi:10.1016/0020-0190(81)90111-3.
- 13 Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528, March 2003. doi:10.1109/TKDE.2003.1198387.
- 14 Kamal Jain and Vijay Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of The ACM - JACM*, 48, March 2001. doi:10.1145/375827.375845.
- 15 Shaofeng H.-C. Jiang, Erzhi Liu, You Lyu, Zhihao Gavin Tang, and Yubo Zhang. Online facility location with predictions. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. arXiv:2110.08840.
- 16 Haim Kaplan, David Naori, and Danny Raz. Almost tight bounds for online facility location in the random-order model. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 1523–1544. SIAM, 2023. doi:10.1137/1.9781611977554.ch56.
- 17 Jyh-Han Lin and Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, 44(5):245–249, 1992. doi:10.1016/0020-0190(92)90208-D.
- 18 Adam Meyerson. Online facility location. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001. doi:10.1109/SFCS.2001.959917.
- 19 Andrew Suk and Ji Zeng. A Positive Fraction Erdős-Szekeres Theorem and Its Applications. In Xavier Goaoc and Michael Kerber, editors, *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 62:1–62:15, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.SoCG.2022.62.

A Randomized Algorithm for Online Facility Location

Here, we mention the algorithm, RFL, as suggested by Meyerson[18], to illustrate the motivation behind our algorithms.

■ **Algorithm 2** Randomized Facility Location (RFL).

```

1: procedure  $RFL(x_i)$  ▷ Input request is  $x_i$ 
2:    $F_i \leftarrow \text{Nearest facility to } x_i$ 
3:    $p_i \leftarrow \min\{1, \frac{d(x_i, F_i)}{f}\}$ 
4:   With probability  $p_i$  open a new facility at  $x_i$  and allocate  $x_i$  to it
5:   With probability  $1 - p_i$ , assign  $x_i$  to  $F_i$ 
6: end procedure

```

B Facility Location with Congestion (with proofs)

In this section, we restate our results on congestion and provide the necessary proofs. First, let us recall Definition 3.1, where we defined Online Facility Location with Congestion. Also, note that in this paper, we consider congestion costs which satisfy $g(a \cdot b) = g(a) \cdot g(b)$ (for example a monomial). For this particular model, we prove the following :

► **Theorem B.1.** *In the online facility location problem with congestion,*

1. *There exists an online algorithm attaining a competitive ratio of $\mathcal{O}(\frac{\log k^*}{\log \log k^*})$ in expectation,*
where $k^ := 2 \cdot g^{-1}(\frac{f}{g(2)-2})$, a constant independent of the number of requests.*
2. *Furthermore, no randomized online algorithm can achieve a competitive ratio better than $\frac{\log k^*}{\log \log k^*}$.*

In the rest of this section, we first prove the Lower Bound for the Worst Case input in Appendix B.1. Then we prove that the same bound is achieved, in expectation, by a randomized algorithm in Appendix B.2. However before we do these, we shall make a key observation that in the congestion model, an offline optimal algorithm will not allocate too many requests to a single facility.

Let k' be the smallest integer such that $g(k' + 1) - g(k') \geq f$. If an online algorithm allocates $k' + 1$ requests to one facility then the congestion cost for the $k' + 1$ -th request itself is $g(k' + 1) - g(k') \geq f$. So no online algorithm needs to put more than k' requests in a facility as it might as well open a new facility and incur less cost.

The above reasoning holds for offline algorithms too and specifically for the offline optimal algorithm, OPT . However we can give an even better bound when considering OPT . For this we shall assume that the congestion function satisfies $g(a \cdot b) = g(a) \cdot g(b)$.

▷ **Claim B.2.** Any facility opened by OPT has at most $k^* := 2 \cdot g^{-1}(\frac{f}{g(2)-2})$ requests allotted to it.

Proof. Let F be a facility opened by OPT with k requests in it. Since OPT did not split the k requests into two facilities but kept them in a single facility, it must be the case that splitting them into two facilities costs more. Notice that the cost paid if two facilities are opened is $\geq 2f + g(a_1) + g(a_2)$ in addition to some distance cost, where $a_1 + a_2 = k$. Since g is a convex non-decreasing function, we have $g(a_1) + g(a_2) \geq 2g(\frac{k}{2})$ (Here we have assumed that k is even, else we can look at $k - 1$ instead of k). Also opening two facilities means that the total distance cost is lesser compared to opening only one facility. Hence we can conclude $2f + 2g(\frac{k}{2}) \geq f + g(k)$. Since we are working under the assumption that $g(a \cdot b) = g(a) \cdot g(b)$, we can now write $f \geq (g(2) - 2)g(\frac{k}{2}) \implies \frac{k}{2} \leq g^{-1}(\frac{f}{g(2)-2}) \implies k \leq 2 \cdot g^{-1}(\frac{f}{g(2)-2})$. ◁

B.1 Lower Bound

Here we shall present a proof of Theorem B.1-Item 2. The proof will be along the same lines as the previous lower bound proof.

Proof of Theorem B.1-Item 2. Consider a binary tree of depth h . The distance between root node and its children is $\frac{f}{h}$ but the distance keeps decreasing as we traverse down the tree. The distance between any level i node and its child will be $\frac{f}{h^{i+1}}$, where we will decide later what h to choose. The input will comprise of 1 request at root node, followed by h requests at a node at level 1, and so on, with h^i requests for some node at the i th level. The requested nodes will form a path from the root node to a leaf node, say ℓ . Also there will be a total of $n = 1 + h + h^2 + \dots + h^{h-1} = \frac{h^h - 1}{h - 1}$ requests.

One may use this sequence of requests to obtain the required lower bound. Due to lack of space, the proof has been skipped here. The full version of the paper [5] contains the complete proof. ◀

B.2 Upper Bound

We have already seen a lower bound of the online facility location problem with congestion. Now we will present an algorithm to solve the online facility location problem with congestion and study the upper bound. We shall present an algorithm that asymptotically attains the same competitive ratio as suggested by the lower bound, making our analysis tight.

We shall present this algorithm as a modified version of *RFL*. Then we shall study the modified algorithm and try to compare the competitive ratio of the modified algorithm with congestion to the competitive ratio of the original algorithm without congestion.

However notice that the lower bounds for the competitive ratio is different for the case with congestion as compared to the original case without congestion. As we have seen earlier, the lower bound for the competitive ratio is $\frac{\log k^*}{\log \log k^*}$, when we are looking at online facility location with congestion. However in the original setting without congestion, the lower bound was $\frac{\log n}{\log \log n}$, which as we have seen is attained by *RFL* in expectation. Intuitively, the reason why we will be able to go from n to k^* is because in the congestion model we have the guarantee that *OPT* will not have more than k^* requests allocated to any facility so if we take one facility opened by *OPT* and consider the requests allotted to it, we can expect to somehow get a competitive ratio of $\frac{\log k^*}{\log \log k^*}$. After that we will just have to show that the competitive ratio holds true even when there are multiple facilities opened by *OPT*.

We shall use the algorithm *RFL* as a black box in order to obtain this modified algorithm, which we shall name *MRFL*.

■ **Algorithm 3** Modified RFL (MRFL).

```

1: procedure MRFL( $x_i$ )                                     ▷ Input request is  $x_i$ 
2:   On input request  $x_i$ , run algorithm RFL on the current state for the input  $x_i$  and
   perform the same action.
3:   if Any facility,  $F$ , has  $k^*$  requests allocated to it then
4:     “Close” facility  $F$ 
5:     Open a new facility at the same location as  $F$ 
6:   end if
7: end procedure

```

We shall now prove Theorem B.1-Item 1, in particular for the Algorithm *MRFL*.

Proof of Theorem B.1-Item 1. Firstly, by closing a facility, we mean that no more requests will be allocated to the facility anymore by the algorithm. No change in the model is assumed since the facility still exists: just that the algorithm will not allocate any more requests to it. Since another facility is opened right after a facility closes, at the same exact location, for future queries the algorithm will have this new facility available. Therefore the actions taken by *RFL* in the model without congestion (For e.g., allocating a request to a facility which has now become “closed”) can always be performed by *MRFL* in the model with congestion (Since the “closed” facility has another facility open at the exact same location). Also apart from the extra facilities opened by *MRFL* in step 5, the actions taken by *RFL* and *MRFL* are identical. This means that the distance cost paid by the algorithm *RFL* on a model without congestion is the same as the distance cost paid by the algorithm *MRFL* on a model with congestion whenever the sequence of input requests is the same. We cannot say the same for the facility opening cost though, since the algorithm *MRFL* opens excess facilities in step 5. However if we separate the facilities opened by *MRFL* into two categories : facilities opened by *MRFL* in step 2 and facilities opened by *MRFL* in step 5, we will notice that the number of facilities opened by *MRFL* in step 2 is equal to the number of facilities that *RFL* would have opened in the congestion free model for the same sequence of input requests. Visually, the total number of points in the metric space where a facility is opened is same for *MRFL* and *RFL*, just that *MRFL* might have multiple facilities opened at a few points in the metric space. Also the congestion cost is something that only *MRFL* has to pay and *RFL* does not need to pay since *RFL* runs on a congestion free model.

Based on this observation, we shall now try to divide the total cost paid by *MRFL* into two parts. Let C be the total cost paid by *MRFL* over the entire sequence of input requests. Let C_1 be the total distance cost paid by *MRFL* plus the total facility opening cost paid by *MRFL* for facilities opened in step 2 of the algorithm. Let C_2 be the total congestion cost paid by *MRFL* plus the total facility opening cost paid over all facilities opened in step 5 of the algorithm.

Clearly $C = C_1 + C_2$ and hence we can write the competitive ratio of algorithm *MRFL* as

$$\mu = \frac{C}{C_{OPT}} = \frac{C_1 + C_2}{C_{OPT}}.$$

where C_{OPT} is the cost paid by an optimal offline algorithm on the same sequence of input requests.

▷ **Claim B.3.** If the total number of input requests is n then the cost C_2 incurred by algorithm *MRFL* is at most $\frac{n}{k^*} f \left(1 + \frac{g(2)}{g(2)-2} \right)$.

Due to lack of space, the proof of this claim has been omitted. It can be found in the extended version of this paper [5].

At this point, using Claim B.2, we can say the OPT must open at least $\frac{n}{k^*}$ facilities and hence $C_{OPT} \geq \frac{n}{k^*} f$. Also using Claim B.3, we know that $C_2 \leq \frac{n}{k^*} f \left(1 + \frac{g(2)}{g(2)-2} \right)$. Therefore,

$$\frac{C_2}{C_{OPT}} \leq 1 + \frac{g(2)}{g(2)-2}. \quad (1)$$

Hence we can focus our attention on $\frac{C_1}{C_{OPT}}$, since $\frac{C_2}{C_{OPT}}$ is a constant. Let us first try to understand what these terms represent. C_1 represents the total distance cost paid by the algorithm *MRFL* plus the facility opening cost for facilities opened in step 2 of the Algorithm 3. We notice that the distance cost paid by *MRFL* over a sequence of input requests is equal to the distance cost paid by *RFL* over the same sequence of input requests

in the model without congestion. Also, the number of facilities opened by *RFL* is the same as the number of facilities opened by *MRFL* in step 2 by construction. Hence C_1 , in expectation, is the cost paid by *RFL* on the same sequence of input requests in the model without congestion.

However the denominator C_{OPT} is the cost paid by *OPT* for the same sequence of input requests in the model with congestion. One thing to note here is that if the denominator was the cost paid by *OPT* without congestion, we would have been stuck because then this ratio would have been $\mu(n) := \frac{\log n}{\log \log n}$, which is much larger than the competitive ratio of $\frac{\log k^*}{\log \log k^*}$ we are trying to achieve.

At this point we would have ideally liked to say that since *OPT* allocates at most k^* requests to any facility, let us take any particular facility c^* opened by *OPT*. If we looked only at the requests allotted to c^* by *OPT*, the cost paid by algorithm *RFL* is at most $\mu(k^*)$ times the cost paid by *OPT* had the input been only these requests allotted to c^* by *OPT*. However the input actually comprises of several such clusters, where each cluster is the set of requests that were allocated to the same facility by *OPT*. While it is true that the competitive ratio would have been $\mu(k^*)$ if the input was only one such cluster, it is not clear if that will be the case for the entire input sequence. This is because while *OPT*'s cost over the entire input sequence is exactly the sum of costs of the clusters, *RFL* might end up paying more on the entire input when compared to *RFL*'s costs on the clusters individually (had the input been just the cluster), summed up.

More formally, let us view the entire input sequence of length n as clusters of input sequences where each such input sequence corresponds to the requests allotted to a given facility. That is $n = \sum_j a_j$, where a_j is the number of requests allotted to the j -th facility. Also, we know that $a_j \leq k^*$ for all j . Additionally had the input been just the a_j requests, $\frac{C_1}{C_{OPT}}$ would have been less than $\mu(k^*)$. Now that the various clusters are given as input together, the cost paid by *OPT* is exactly the sum of the costs of the clusters separately. However for the online algorithm *RFL* the cost paid on the entire input might be more than the sum of the costs over the clusters (assuming that the input was just the cluster). Our initial guess was that this cost C_1 over the entire sequence might still be less than some constant times the sum of the costs of the clusters individually. However we have been unable to prove this.

We observe that the denominator has *OPT*'s cost in the model with congestion. So *OPT* would have at most k^* requests allotted to any facility (Using Claim B.2). Let us now concentrate on any facility opened by *OPT* and look at the requests that are allocated to the facility. So let us write $S = S_1 \cup S_2 \cup \dots \cup S_k$, where S is the entire collection of input requests and S_i is the set of input requests that were allotted to a particular facility and k is the total number of facilities opened by *OPT*. Let C_{S_i} represent the cost paid by *OPT* in the model with congestion if the input was just S_i instead of S and let \tilde{C}_{S_i} represent the cost paid by *OPT* if the input was S_i in the model without congestion. Similarly let $C_{1_{S_i}}$ be the cost paid by *RFL* (recall that for this we are only looking at the model without congestion) over the requests in S_i but when the input is the entire input S .

$$\frac{C_1}{C_{OPT}} = \frac{C_1}{\sum_i C_{S_i}} \leq \frac{C_1}{\sum_i \tilde{C}_{S_i}} = \frac{\sum_i C_{1_{S_i}}}{\sum_i \tilde{C}_{S_i}} \leq \max_i \left\{ \frac{C_{1_{S_i}}}{\tilde{C}_{S_i}} \right\}. \quad (2)$$

Here the denominator, \tilde{C}_{S_i} is the cost paid by *OPT* if the input was S_i in the model without congestion. However the numerator is $C_{1_{S_i}}$, which is the cost paid by *RFL* in the model without congestion when the input is the entire input S . Had the numerator been the cost paid by *RFL* in the model without congestion when the input is just S_i , we would have stated that this is exactly $\mu(|S_i|)$. Also since $|S_i| \leq k^*$ for all i , we would have obtained $\frac{C_1}{C_{OPT}} \leq \mu(k^*) = \mathcal{O}\left(\frac{\log k^*}{\log \log k^*}\right)$ for *RFL*.

The only problem is that $C_{1_{S_i}}$ is the cost paid by RFL , in the model without congestion when the input is the entire input S . This means that RFL might have extra facilities already open when the requests of S_i start coming in. Also new facilities might open between any two requests of S_i since other requests from S might be interleaved in between requests from S_i . The analysis of RFL in [9] actually goes through if there were open facilities available before the first request comes and also if new facilities appeared in between two requests (Interestingly, any algorithm that satisfies this property, could have been used as a black box instead of RFL). For the sake of completion, we add a thorough analysis of this in the full version of the paper [5]. This completes the proof that the competitive ratio of $MRFL$ is $\frac{\log k^*}{\log \log k^*}$. ◀

C Missing Proofs

Proof of Property 5.7. Let $X_i = \{(x_i, w_i), (x_{i+1}, w_{i+1}), \dots, (x_m, w_m)\}$.

Let C_i be the expected cost paid by $WRFL$ before a facility opens at any request in X_i , when the input is X_i . Note that now, we are interested in C_1 , which is the expected cost paid by $WRFL$ before a facility opens in X . From Algorithm 1, we recall that $WRFL$ opens a new facility at the request x_i with probability p_i and with the remaining probability it assigns x_i to the nearest open facility.

One can see that $C_1 = p_1 \cdot 0 + (1 - p_1)(p_1 \cdot f + C_2)$ since with probability p_1 a facility is opened at x_1 itself and no cost is incurred before the first facility is opened but with probability $(1 - p_1)$, no facility is opened and a cost of $p_1 \cdot f = d(x_1, F) \cdot w_1$ is incurred. However since we have not yet opened any facility, in expectation C_2 additional cost will be incurred before a facility is opened - because of how C_2 is defined.

We can use the same argument for any C_i now. That is, when X_i is given as input, with probability p_i a facility is opened at x_i itself and no cost is incurred before the first facility is opened and with probability $(1 - p_i)$, no facility is opened and a cost of $p_i \cdot f$ is incurred

$$C_i = (1 - p_i)(p_i \cdot f + C_{i+1}).$$

Now notice that C_m is the expected cost paid by $WRFL$ before a facility opens when input is $\{(x_m, w_m)\}$. Therefore $C_m = p_m \cdot f \cdot (1 - p_m)$ since with probability $(1 - p_m)$, facility is not opened and then the cost paid is $p_m \cdot f$. This means that $C_m \leq f$.

Assuming $C_{i+1} \leq f$ for some i , $C_i \leq (1 - p_i)(p_i \cdot f + f) = f(1 - p_i)(1 + p_i) = f(1 - p_i^2) \leq f$.
 $\therefore C_1 \leq f$. ◀

Proof of Property 5.8. Since the distance of x_i to the nearest open facility is $d(x_i, F)$, $WRFL$ would open a new facility with probability $\leq \frac{d(x_i, F) \cdot w_i}{f}$ and pay a cost f and with remaining probability (of at most 1) it would pay a cost of $d(x_i, F) \cdot w_i$. Therefore expected cost paid for the request x_i is at most

$$\frac{d(x_i, F) \cdot w_i}{f} \times f + 1 \times d(x_i, F) \cdot w_i = 2d(x_i, F) \cdot w_i. \quad \blacktriangleleft$$