Online Facility Location with Deletions

Marek Cygan
Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
cygan@mimuw.edu.pl

Artur Czumaj
Department of Computer Science and Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Coventry CV4 7AL, United Kingdom
A.Czumaj@warwick.ac.uk

Marcin Mucha
Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
mucha@mimuw.edu.pl

Piotr Sankowski
Institute of Informatics, University of Warsaw, Banacha 2, 02-097 Warsaw, Poland
sank@mimuw.edu.pl

Abstract

In this paper we study three previously unstudied variants of the online Facility Location problem, considering an intrinsic scenario when the clients and facilities are not only allowed to arrive to the system, but they can also depart at any moment.

We begin with the study of a natural *fully-dynamic online uncapacitated* model where clients can be both added and removed. When a client arrives, then it has to be assigned either to an existing facility or to a new facility opened at the client's location. However, when a client who has been also one of the open facilities is to be removed, then our model has to allow to reconnect all clients that have been connected to that removed facility. In this model, we present an optimal $O(\log \mathfrak{n}_{\mathfrak{act}}/\log\log \mathfrak{n}_{\mathfrak{act}})$-competitive algorithm, where $\mathfrak{n}_{\mathfrak{act}}$ is the number of active clients at the end of the input sequence.

Next, we turn our attention to the *capacitated* Facility Location problem. We first note that if no deletions are allowed, then one can achieve an optimal competitive ratio of $O(\log \mathfrak{n}/\log\log \mathfrak{n})$, where $\mathfrak{n}$ is the length of the sequence. However, when deletions are allowed, the capacitated version of the problem is significantly more challenging than the uncapacitated one. We show that still, using a more sophisticated algorithmic approach, one can obtain an online $O(\log \mathfrak{m} + \log \mathfrak{c} \log \mathfrak{n})$-competitive algorithm for the capacitated Facility Location problem in the fully dynamic model, where $\mathfrak{m}$ is number of points in the input metric and $\mathfrak{c}$ is the capacity of any open facility.

2012 ACM Subject Classification Theory of computation - Online algorithms

Keywords and phrases online algorithms, facility location, fully-dynamic online algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.21

Funding Research partially supported by the Royal Society International Exchanges Scheme 2013/R1, grant IE130346, by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC awards EP/D063191/1, EP/N011163/1, and by the European Research Council research and innovation programme (ERC grant agreement No 677651).

1 Introduction

The Facility Location is one of the central combinatorial optimization problem, extensively studied in the literature for several decades, see, e.g., [5, 12, 14, 29, 37, 45] and the references therein. The goal is to connect a given set of clients to a set of facilities such that the service cost is optimized. A very natural setting for the Facility Location problem is the *online* scenario, where clients arrive incrementally over time and need to be connected to existing or newly opened facilities. Indeed, since it has been introduced over 15 years ago by Meyerson [40], the online version of the Facility Location problem and its generalizations received considerable amount of attention [3, 6, 7, 16, 17, 20, 21, 22, 25, 26, 33, 41, 42, 43, 48]. Typically in these models one assumes to be given a metric over a set of candidate points. When a client appears, we are allowed to open a new facility in her location, paying some cost for opening a new facility, and then we have to irrevocably connect the client to one of the open facilities, paying for the connection the cost equal to the distance from the client to the facility. In this paper, we study a more complex scenario and consider a natural extension of this model allowing clients to be *removed* from the system, extending the standard dynamic model to the *fully dynamic* setting. That is, in each time step either a new client arrives and needs to be connected to one of the open facilities, or one of the clients already existing in the system departs. Observe that if a client who has been also one of the open facilities is to be removed, then our model has to allow to reconnect all clients that have been connected to that removed facility.

The fully dynamic model is very natural in the context of the Facility Location problem, where in a number of scenarios it is desirable to dynamically process the arrival objects, and then to allow their departures. For example, if one wants to build and maintain schools in a newly developing city, one wants to allow a steady arrival of new pupils to the area, and also allow changes in the school demands when pupils population is declining. Similarly, if one wants to maintain a construction of a network, where all clients are to be connected to the servers (and pay connection costs) and each client is allowed to host a server (and pay an opening cost), one may also want to allow the removal of some clients and with that also a closure of some servers. Given that the relocation costs are often very expensive, the decisions should be regarded as irrevocable, unless, as in the case of deletions, are necessary. The framework considered in this paper is even more natural in complex distributed settings, for example, in some more modern scenarios that have been recently motivated by applications in peer-to-peer systems, when client and facility are coupled to the same entity. Consider the so called p2p networks with super-peers [49] that were used for filesharing as Gnutella [30] or can be used for decentralized online social networks [44], distributed game systems [47], grid management systems [19] and distributed storage [34]. In such case, some of the nodes decide to host the provided content. This decision incurs some cost to this node, but can reduce the cost of serving other nodes. The main question asked about such systems is about the needed ratio of superpeers to peers that guarantees that enough capacity of superpeers is available to serve all clients [49]. This measure corresponds exactly to competitive ratio in our models.

Before discussing our results and techniques further, let us first formally define the models starting with the classical online variant.

1.1 The model

We study the performance of online algorithms for the Facility Location problem in the standard framework of *competitive analysis* (cf. [11, 46]). A randomized algorithm for the

Facility Location problem is α-*competitive* if for any input sequence, its expected cost is at most α times the optimal cost for the corresponding instance of the offline Facility Location. Note that the corresponding instance of the offline problem contains only the clients that are active at the end of the input sequence. We consider a standard special version of the Facility Location problem, where the set of clients and the set of possible facility locations are identical (see, e.g., [8, 23, 40]).

1.1.1 *Online* facility location

We consider the Facility Location problem, where points (from some metric space) arrive in online fashion. When a point x arrives, we can first open a facility in x and then we have to assign x to some open facility. These choices are irrevocable. We consider only the Facility Location problem with *uniform opening costs*, which are all, without loss of generality, equal to 1. The restriction to the study of uniform costs makes the problem interesting, since without this assumption, no bounded competitive ratio can be obtained (as described in the full version of this paper).

The cost of an obtained solution for a point set $\mathcal{X}$ is:

$$|\mathcal{F}| + \sum_{x \in \mathcal{X}} dist(x, \mathfrak{a}(x)) \ ,$$

where $\mathcal{F}$ is the set of open facilities and $\mathfrak{a}(x)$ is the open facility to which x is assigned. The cost of a point $x \in S$ is equal to its *connecting cost* (distance to the open facility assigned to x) plus its *opening cost* (if x is an open facility then the opening cost is 1, and it is 0 otherwise).

The model defined here has been studied in the past, see e.g., Fotakis [23] for a survey or Meyerson [40]. (One frequently assumes (see e.g., [40]) that the opening cost in any facility is equal to some f, but simple scaling makes this problem equivalent to the one studied in our paper, that is, with $f = 1$.)

1.1.2 Online facility location with *deletions*

In this paper, we consider a *fully-dynamic online* setting, in which in each time step either a new point (from some metric space) arrives in an online fashion, or a point already in the input is deleted. To cope with the case when an open facility is deleted, we have to allow the input points to be reassigned, and possibly to open other facilities. We consider the following, very natural model:

- In the online process, the requests are arriving online, and each request is either an arrival of a new demand point from a metric space, or a request to remove a previously inserted demand point.
- When a new demand point x arrives, the algorithm must at once and irrevocably decide if a new facility will open at x, and then must assign x to some open facility (possibly to itself, if a facility was open at x).
- When a point x is requested to be removed, then x is deleted from the system and if x was an open facility, then all points assigned to x will be immediately reassigned to other facilities and some of these points may become open facilities.

1.1.3 Online *capacitated* facility location

We also consider a more general model of online *capacitated* Facility Location with deletions, in which each open facility can handle at most $\mathfrak{c}$ clients, that is, at any moment at

most $\mathfrak{c}$ clients can be connected to any single facility. Again, we study only a *uniform* case (in which each facility has the same capacity $\mathfrak{c}$), since otherwise no bounded competitive ratio can be obtained (see full version of the paper).

The goal of an algorithm for the Facility Location in any of the models defined above is to minimize the cost of the solution and to obtain an algorithm that is α-competitive for α as small as possible. The performance of the algorithm may be a function of the *length of the input sequence* $\mathfrak{n}$, the *number of points* $\mathfrak{n}_{\mathtt{act}}$ *active* in a given moment, the *size of the input metric space* $\mathfrak{m}$, and the *capacity* $\mathfrak{c}$.

We note that in the model of uncapacitated Facility Location, the known results (as well as our new results) work for any metric space, even if it is *unknown to the algorithm*. However, for the new algorithm for online capacitated Facility Location with deletions (cf. Theorem 4.4), we will assume that *the underlying metric is known to the algorithm in advance*.

1.2 Related work

As one of the fundamental problems in operations research and combinatorial optimization, the Facility Location problem has been studied extensively in the past, see, e.g., the standard exposition in [14] and more recent advances in [4, 5, 29, 45], and the references therein. In the online setting, an early research focused on the k-median problem (see, e.g., [39]), which is a variant of the Facility Location problem where exactly k facilities have to be opened. Soon after, Meyerson [40] designed a simple randomized online algorithm in the uncapacitated model without deletion. His online algorithm is $O(\log \mathfrak{n}/\log\log \mathfrak{n})$-competitive, where $\mathfrak{n}$ is the number of points in the input. (In fact, only a competitive ratio of $O(\log \mathfrak{n})$ was proven in [40], but Fotakis [22] extended the analysis from [40] to obtain a competitive ratio of $O(\log \mathfrak{n}/\log\log \mathfrak{n})$.) Fotakis [22] later has shown that this bound is asymptotically tight and no online algorithm is $o(\log \mathfrak{n}/\log\log \mathfrak{n})$-competitive; the lower bound holds for randomized algorithms against the oblivious adversary, for uniform facility costs, and for very simple metric spaces, such as the real line. For more discussion about the history of the online version of the uncapacitated Facility Location problem (including deterministic online algorithms and incremental online algorithms), we refer to a survey by Fotakis [23].

The extension of the online model to deal with deletion of the facilities makes the Facility Location problem significantly more challenging. While this model is very natural, it requires a different approach that must permit to *reverse some* of the decisions in the online algorithm, and we are not aware of any study of algorithms for the online Facility Location problem in the fully dynamic setting, where deletions are allowed.

We note however, that similar fully dynamic models for optimization problems on graphs have been considered in the past, though only in limited settings. For example, a number of graph optimization problems have been considered in fully dynamic models in the setting of *data streams*, in the so-called *turnstile model*. This model has been investigated in two scenarios: in the context of geometric graph optimization problems (see, e.g., [15, 28, 35]), and only very recently, in the context of standard graph optimization problems (see, e.g., a recent survey [38] and the references therein). The main focus of these studies is to design algorithms that process a stream of data (in this case, edge or vertex insertions and deletions) and *using very limited space*, to maintain some basic graph features. For *geometric graph optimization problems*, where the input is defined over a set of points in the discrete d-dimensional space $\{1, 2, \ldots, \Delta\}^d$, it has been shown that many basic properties (e.g., k-

median, minimum spanning tree, minimum weight matching, MaxCut) can be approximated very efficiently even with poly-logarithmic space (see, e.g., [28, 24]). The uncapacitated Facility Location problem has been studied in the context of data streaming, initiated with work of Indyk [28], who gave a poly$(\log \Delta)$-space streaming algorithm that approximates the optimal cost of the Facility Location problem within a factor of $O(\log^2 \Delta)$. The best currently known streaming algorithm using poly$(\log \Delta)$-space gives an $(1+\varepsilon)$-approximation for this problem [15]. The research in data streaming for *standard graph optimization problems* has been traditionally focusing on the insertion-only model, where one was aiming to design streaming algorithms with $O(n \operatorname{poly} \log n)$ space (cf. [38] and the references therein). Only very recently, Ahn et al. [1] initiated the study of algorithms that allow both insertions and deletions (see also [13]). This line of research led to a number of efficient data streaming algorithms for fundamental graph problems, such as testing connectivity or bipartiteness, computing spanning trees and various graph sparsifiers, maximum matching, that can be (approximately) computed in small, $O(n \operatorname{poly} \log n)$ space not only in the insertion only model, but also in the model with deletions [1, 2, 31, 32, 38].

We also note that recently there has been some research on the standard online Steiner tree problem with deletions, see e.g., [27, 36].

1.3 New results

The main contribution of this paper is the first thorough study of the online Facility Location problem with deletions and design of new algorithms for this model in several natural settings.

We begin with the study of the simplest, *uncapacitated* model. We present in Theorem 2.2 an online $O(\log \mathfrak{n}_{\mathtt{act}} / \log \log \mathfrak{n}_{\mathtt{act}})$-competitive algorithm for the uncapacitated Facility Location problem with deletions, where $\mathfrak{n}_{\mathtt{act}}$ is the number of active clients at the end of the input sequence; this bound gives an asymptotically optimal competitive ratio. Our algorithm is an extension of the classical insertion-only algorithm for the Facility Location problem due to Meyerson [40], and we show that one can modify the analysis from [40] to allow deletions.

Next, we turn our attention to the *capacitated* Facility Location problem. We first prove (Observation 3.1) that if *no deletions* are permitted, then a simple modification of Meyerson's algorithm for online Facility Location can be applied to achieve an optimal competitive ratio of $O(\log \mathfrak{n} / \log \log \mathfrak{n})$. However, when deletions are allowed, then the capacitated version of the problem is significantly more challenging than the uncapacitated one. We show that still, using more involved approach incorporating hierarchically well-separated trees, one can obtain an online $O(\log \mathfrak{m} + \log \mathfrak{c} \log \mathfrak{n})$-competitive algorithm for the capacitated Facility Location problem with deletions, in the fully dynamic model, where $\mathfrak{n}$ is the number of queries, $\mathfrak{m}$ is the number of points in the input metric, and $\mathfrak{c}$ is the capacity of the facilities (Theorem 4.4).

We notice that while the algorithms from Theorem 2.2 and Observation 3.1 do not need to know the input metric, the result from Theorem 4.4 assumes that the input metric is known to the algorithm.

Our work demonstrates that despite the fact that the online Facility Location problem with deletion is clearly more complex than the classical online problem with insertions only, the most natural variants of these problems, for both the uncapacitated and the capacitated model for uniform facility costs, have very efficient online algorithms that achieve competitive ratios matching or almost matching those of the insertion only variants of the problem.

2 Online uncapacitated facility location

We begin with the study of the simplest, *uncapacitated* model, and describe an insertion-only online algorithm for uncapacitated Facility Location due to Meyerson [40].

Algorithm M:

When a new demand point x arrives then find its nearest open facility y and set $d_x = \min\{\mathsf{dist}(x,y), 1\}$. Next, with probability d_x open a new facility at point x and assign x to it; otherwise, assign x to y.

Meyerson [40] proved that Algorithm M is $O(\log \mathfrak{n}/\log\log \mathfrak{n})$-competitive in the model with insertions only (see also [22, 23]).

The most natural approach to obtain an online algorithm for the Facility Location problem with deletions is to attempt to modify Algorithm M. Indeed, there is a simple modification of Algorithm M that addresses the deletions. The following Algorithm M* proceeds as in Algorithm M, except that when a facility is removed, it reprocesses all the points that are now not assigned to any facility using the original algorithm.

Algorithm M*:

When a new demand point x arrives then find its nearest open facility y and set $d_x = \min\{\mathsf{dist}(x,y), 1\}$. Next, with probability d_x open a new facility at point x and assign x to it; otherwise, assign x to y.

When a point x that is an open facility is to be deleted, then reassign all points assigned to x using the algorithm for the insertions.

While it is appealing to hope that Algorithm M* has performance similar to that of Algorithm M for insertions only, but in fact asymptotically, it does no better than opening facilities at all points.

▶ **Claim 2.1.** Algorithm M* has competitive ratio of $\Omega(\mathfrak{n}_{\mathfrak{act}})$.

Proof. For any $k \in \mathbb{N}_+$, consider a star metric with center $\mathfrak{o}$ connected to points $\mathcal{X} = \{x_1, \ldots, x_k\}$ at distance $\varepsilon = \frac{1}{k}$ from $\mathfrak{o}$. Consider the following input sequence:

1. Add k^2 clients $a_1, \ldots, a_{k^2}$ located at $\mathfrak{o}$.
2. For each $i = 1, \ldots, k$ add a client b_i located at x_i.
3. Remove the clients $a_1, \ldots, a_{k^2-1}$ in that order.

We will analyze the performance of algorithm M^* in the above scenario. While doing that, we assume that whenever clients are reassigned, they are reassigned in the order in which they were originally assigned.

Consider the first two stages, when all the clients are added. In step 1, client a_1 opens a facility and then all other clients at $\mathfrak{o}$ connect to a_1. In step 2, each b_i opens a facility with probability ε and connects to a_1 otherwise. For the remainder of the analysis, let us assume that at least one of the clients b_j opens a facility. This happens with probability $p_1 = 1 - (1-\varepsilon)^k = \Omega(1)$.

Next, we remove a_1 and reassign all other clients assigned to a_1. One by one, each client at $\mathfrak{o}$ flips a coin and with probability ε connects to one of the b_i's; otherwise it opens a facility. As soon as some a_i opens a facility, all other clients at $\mathfrak{o}$ connect to a_i, and each client at $\mathcal{X}$ that has not yet opened a facility (and hence was initially connected to a_1) does so with probability ε; otherwise it connects to a_i. If no a_i opens a facility, then each client

at $\mathcal{X}$ that has not yet opened a facility, opens one with probability 2ε, and otherwise it connects to one already open facility at some b_j.

We then remove all other clients $a_2, \ldots, a_{k^2-1}$. Each time the client removed has a facility opened, the scenario described in the previous paragraph repeats. Removal of a client with no facility opened has, of course, no effect on the other clients.

Note that each time we remove a client at $\mathfrak{o}$ that has a facility open, each of the b_j's opens a facility with probability at least ε, unless it has already opened a facility. How many times does that happen? Let us count the number of a_i's with $2 \leq i \leq k^2 - 1$ that at some moment open a facility (and then get removed). Each of the a_i's flips a coin exactly once (which corresponds to the situation that at some moment we have already removed $a_1, \ldots, a_j$ for some $j < i$, and there is no facility open at $a_{j+1}, \ldots, a_{i-1}$, with points $a_{j+1}, \ldots, a_{i-1}$ connected to facilities from $\mathcal{X}$). Furthermore, the coin flip for a_i is independent from coin flips made by $a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{k^2-1}$. Therefore the number of a_i's that at some moment open a facility and then get removed has the binomial distribution with parameters $k^2 - 2$ and ε. Hence, by Chernoff bound, with probability $p_2 \geq 1 - e^{-\varepsilon(k^2-2)/6} = \Omega(1)$, there are at least $\varepsilon(k^2-2)/2 \geq k/2-1$ of those clients. Therefore, we can conclude that each b_j opens a facility with probability at least $p_3 = 1-(1-\varepsilon)^{k/2-1} = \Omega(1)$.

The optimal offline solution opens a facility at a_{k^2} and connects all points $b_1, \ldots, b_k$ to a_{k^2}, and it has cost $1 + \varepsilon k = 2$. On the other hand, with probability $p_1 p_2 = \Omega(1)$ algorithm M^* opens at least $kp_3 = \Omega(k)$ facilities from $b_1, \ldots, b_k$ in expectation. This gives an $\Omega(k) = \Omega(\mathfrak{n}_{\mathfrak{act}})$ competitive ratio. So asymptotically M^* does no better than just opening facilities at all points. ◀

2.1 Asymptotically optimal competitive ratio for uncapacitated facility location with deletions

Claim 2.1 shows one of the main challenges that online algorithms with deletions must cope with: we cannot let points attempt to open a facility too frequently, since then we would open too many facilities, and at the same time we have to be able to open some facilities in the neighborhood of the facilities that we are closing. To address this challenge we have to provide a delicate online strategy that will maintain a right balance between these two desirables.

Algorithm 1:

Newly arriving points: When a new demand point x arrives then find its nearest open facility y and set $d' = \min\{\mathsf{dist}(x, y), 1\}$. With probability d' open a new facility at point x and assign x to it; otherwise, assign x to y and set $p_x := d'$. (The algorithm will memorize p_x for future use.)

Deletion: When a point that is *not an open facility* is to be deleted, then just remove that point.

When a point that is an *open facility* is to be deleted, then reassign all points assigned to it: When a point x is to be reassigned, then find its nearest open facility y and set $d' = \min\{\mathsf{dist}(x, y), 1\}$. Let p_x be the last value used in the processing of x. If $d' \leq 2p_x$, then assign x to y; otherwise with probability d' open a new facility at x; else, with probability $1 - d'$ assign x to y and set $p_x := d'$.

The following theorem analyzes the performance of Algorithm 1.

▶ **Theorem 2.2.** *Algorithm 1 is $O(\log \mathfrak{n}_{\mathtt{act}}/\log\log \mathfrak{n}_{\mathtt{act}})$-competitive, where $\mathfrak{n}_{\mathtt{act}}$ is the number of active clients at the end of the input sequence (in particular, as $\mathfrak{n}$ is the input length, we have $\mathfrak{n}_{\mathtt{act}} \leq \mathfrak{n}$).*

Proof. We follow the approach of Meyerson [40], with special care taken to deal with deletions.

Consider any optimal solution. For a fixed facility v, let S be the set of clients connected to v in the optimal solution. Let d^* be the average distance from the points of S to v. We split S into $h+1$ subsets $S_0, S_1, \ldots, S_h$, where $h = \lceil 2\log|S|/\log\log|S|\rceil$. Points in S_0 are at distance at most d^* from v. For $1 \leq i \leq h$, points in S_i are at distance greater than $d^*(\log|S|)^{(i-1)/2}$, but at most $d^*(\log|S|)^{i/2}$ from v. Note that each point is contained in some S_i, as $d^*(\log|S|)^{h/2} \geq d^*|S|$, and a single client at distance more than $d^*|S|$ would contradict the average distance d^*.

For each set S_i we split the time into two epochs: the second epoch starts when the first point in S_i becomes an open facility in the solution of the algorithm — note that the second epoch might never start.

Consider the first epoch of some S_i. The part of the cost of the final solution incurred by the points in S_i during the first epoch is the total connection cost of these points at the end of the first epoch plus possibly a single opening cost. The connection cost of a point $x \in S_i$ is upper bounded by twice the probability of the last coin flip of x, regardless of whether that connection was preceded by a coin flip or not. To bound the total connection cost of S_i in the first epoch it is therefore enough to bound the sum of the probabilities of all the coin flips related to S_i made during that epoch.

▶ **Lemma 2.3.** *The expected value of the sum of the probabilities of the coin flips related to points in S_i and made before the first facility in S_i opens is at most* 1.

Proof. Let $P_1, P_2, \ldots, P_M$ be the probabilities of all coin flips (with non-zero probabilities) made for points in S_i and let $X_1, X_2, \ldots, X_M$ be the outcomes of these coin flips, so that $P[X_j = 1] = P_j$ (note that each P_j is a random variable, and not a constant since its value might possibly depend on earlier coin flips). Also, add to both sequences a virtual „sentinel" coin flip with $P_{M+1} = 1$, $X_{M+1} = 1$. Define $Z_0 = 0$ and $Z_{j+1} = Z_j + P_{j+1} - X_{j+1}$ for $j = 1, \ldots, M+1$. Then the sequence $\{Z_j\}$ forms a martingale. Let $T = \min\{j > 0 : X_j = 1\}$ be the position of the first heads in $\{X_j\}$. Note, that T is well defined, since $X_{M+1} = 1$. Also note, that $T = \min\{j > 0 : Z_j \leq Z_{j-1}\}$, i.e., T is a stopping time for $\{Z_j\}$. Since T is bounded, from the Doob's optional stopping time theorem we get that $E[Z_T] = E[Z_0] = 0$. However, we also have

$$Z_T = \sum_{j=1}^{T} P_j - 1,$$

and thus the claim follows. Note that the claim does not hold with equality, since the above expression might include the virtual coin flip added to make T well defined. ◀

It follows that the total cost incurred by the points of S_i during the first epoch is a constant, so the overall cost of the first epoch is $O(h)$.

Consider now the cost incurred by any point $x \in S_i$ in the second epoch. If x does not make any coin flips in the second epoch, then any connection made by x has cost upper bounded by $2d^*(\log|S|)^{i/2}$, since there is an open facility in S_i at this point. Consider now the case when x does make a coin flip during the second epoch. We then upper bound the

cost of the last connection made by x by twice the probability of the last coin flip of x. We also need to bound the cost of a facility that x might potentially open. The expected number of facilities opened by x is the sum of the probabilities of all its coin flips. Each term in this sum is at least twice the previous one, and so the expected number of facilities opened by x is at most twice the probability of the last coin flip of x. This probability is at most $2d^*(\log|S|)^{i/2}$ since there is an open facility in S_i.

To sum up, the total cost incurred by $x \in S_i$ during the second epoch is at most $8d^*(\log|S|)^{i/2}$. Consider first the case where S_i is not the innermost layer, i.e., $0 < i \leq h$. Any $x \in S_i$ is then connected to v in the optimum solution, and thus incurs a cost of at least $d^*(\log|S|)^{(i-1)/2}$. Therefore, the cost incurred by such x in the second epoch is at most $\sqrt{\log|S|}$ times the corresponding cost in the optimal solution. Consider now the innermost layer. For any client in S_0 Algorithm 1 pays at most $O(d^*)$, leading to $O(d^*|S_0|)$ total cost of the second epoch, which by the definition of d^* is at most a constant factor more than the connection cost payed by the optimal algorithm.

Note that in the analysis we completely ignore the cost generated by the clients, that are removed during the course of the algorithm, as those clients in the end generate no cost to Algorithm 1. ◀

Since even in the incremental model (when points can only arrive, not vanish) there is an $\Omega(\log \mathfrak{n}/\log\log \mathfrak{n})$-lower bound (cf. [22]), Algorithm 1 is optimal up to a constant factor.

3 Capacitated online facility location (with insertions only)

Our result in Theorem 2.2 shows that for uncapacitated Facility Location with deletions, one can extend the approach from earlier works (see [40] and also [22]) to design online algorithms that achieve asymptotically optimal competitive ratio. However, the model of *capacitated* Facility Location with deletions is significantly more complex. Still, in the most basic case, the model *with insertions only*, we observe that Algorithm M can be extended to the model of capacitated Facility Location. We run Algorithm M with a single modification: the nearest facility y now is the nearest facility that is not fully saturated, that is, that has still available capacity.

▶ **Observation 3.1.** Algorithm M in the capacitated case is $O(\log \mathfrak{n}/\log\log \mathfrak{n})$-competitive in the model with insertions only.

Proof. We only sketch the proof, since it is a straightforward modification of the original proof of Meyerson [40] and Fotakis [22]. Similarly as described in the proof of Theorem 2.2, we partition the set S of clients connected to some open facility v into subsets $S_0, \ldots, S_h$, depending on their distance to v. What is different from the analysis of Meyerson and Fotakis is the way in which we split time into epochs for each layer S_i, as here the first and second epoch can possibly interlace. Formally, the cost incurred by a client belongs to the first epoch, if at the moment when the client appears there is no unsaturated open facility in the layer S_i. If there is an unsaturated open facility in the layer S_i, then the cost incurred by the client is classified to the second epoch.

The total cost of clients of the first epoch is bounded by Lemma 2.3, and is at most $\ell + \mathfrak{n}/\mathfrak{c}$, where ℓ is the total number of sets S_i, as at most $\mathfrak{n}/\mathfrak{c}$ facilities may be saturated during the course of the algorithm. Note that ℓ is exactly $(h+1) = O(\log \mathfrak{n}/\log\log \mathfrak{n})$ times greater than the number of open facilities in the optimal solution. Also, as each facility can serve only $\mathfrak{c}$ clients, the term $\mathfrak{n}/\mathfrak{c}$ is not greater than the cost of the optimal solution.

The cost of clients of the second epoch is bounded as in the proof of Theorem 2.2, i.e., Algorithm M pays at most $d^*(\log|S|)^{i/2}$ for each client from S_i, whereas optimal solution pays at least $d^*(\log|S|)^{(i-1)/2}$, assuming $i > 0$. The cost of clients from S_0 is bounded analogously as in the proof of Theorem 2.2. ◀

4 Capacitated facility location with deletions

The result in Section 3 may give a hope that also our result from Theorem 2.2 for uncapacitated Facility Location with deletions can be easily extended to the model of *capacitated* Facility Location with deletions. For example, let us think of a simple extension of Algorithm 1 to the capacitated case. Perhaps the most natural idea is to let d be the distance to the closest open unsaturated facility, i.e., a facility which still can serve additional clients. Unfortunately this line of reasoning does not lead to a meaningful competitive ratio, because in the case when all the clients arrive at the same location all the distances are equal to zero and therefore such a modified version of Algorithm 1 would be deterministic. Playing against a deterministic algorithm is very convenient for the adversary, as the adversary might remove all the clients which were not turned into open facilities by the algorithm, leading to $\Omega(\mathfrak{c})$ competitive ratio.

One natural idea to introduce randomness to the algorithm is to increase the value of d, so that even if the distance to the closest facility is zero, the client might still decide to open a new facility. However, this idea alone does not seem to lead to any reasonable competitive factor. Below, we present a typical hard example for one possible implementation of this idea. (Note, that this competitive ratio is as bad as the one obtained by an algorithm that opens facilities in all input points.)

▶ **Claim 4.1.** Let $\mathcal{A}$ be Algorithm 1 modified, so that d is the maximum of the distance to the closest unsaturated facility and $10/\mathfrak{c}$. Then, the competitive ratio of $\mathcal{A}$ is $\Omega(\mathfrak{c})$.

Proof. Consider a star metric with the center $\mathfrak{o}$ and the remaining $10\mathfrak{c}^2$ points at distance $\frac{1}{2}$ from $\mathfrak{o}$. Let us analyze the performance of $\mathcal{A}$ on the following input sequence. First, we repeat $\mathfrak{c}$ times the following insertion: insert a point at $\mathfrak{o}$ and then $10\mathfrak{c}$ points in different leaves of the metric, and then, at the end, remove all points located in the leaves.

The optimal solution will have one open facility at one of the $\mathfrak{c}$ points located at $\mathfrak{o}$, and so the optimal cost is 1. We claim that the expected size of the solution found by $\mathcal{A}$ is $\Omega(\mathfrak{c})$.

To see this, consider a single round of insertions. If at the beginning of the round, there are no unsaturated facilities at $\mathfrak{o}$, then one is created with probability $\Theta(1)$. If that happens, then each of the clients inserted at the leaves opens a facility with probability $\frac{1}{2} + 10/\mathfrak{c}$ until the facility is saturated. Since there are $10\mathfrak{c}$ such clients, w.h.p. the facility gets saturated. It also follows, that for any round, w.h.p. there are no unsaturated facilities at $\mathfrak{o}$ at the beginning of the round.

The expected number of facilities opened at $\mathfrak{o}$ is the sum over all rounds of insertions of the probabilities that a facility is open at the beginning of the round. Based on the observations of the previous paragraph, this probability is $\Theta(1)$ for each round, and the claim follows. ◀

4.1 Hierarchically well-separated trees and facility location

We will present an online algorithm for the *capacitated* Facility Location problem in the fully dynamic setting with low competitive ratio, with both insertions and deletions. We begin with a brief overview. We will assume that (unlike in the rest of the paper) *the*

input metric is given in advance, $\mathcal{V}$ is the set of all points in the metric space, and $\mathfrak{m}$ is the number of points in the metric space, $\mathfrak{m} = |\mathcal{V}|$. We use the embedding of the original metric space into hierarchically well-separated trees (cf. [9, 10, 18]), on which we run our Facility Location algorithm. Once we run the algorithm on a hierarchically well-separated tree $\mathfrak{T}$, for every open facility, we will evenly split its capacity into $\mathfrak{h} = O(\log \mathfrak{c})$ parts and then allocate each partial capacity solely to the points in one of $\mathfrak{h}$ areas we will define later. Then we use a key property which ensures that every point v that would use an open facility u in the offline uncapacitated optimal solution, to find a replacement facility with still available capacity that is at the same distance from v in $\mathfrak{T}$ as the distance from u to v in $\mathfrak{T}$.

Our approach relies on the concept of *hierarchically well-separated trees (HSTs)*, which are metric spaces defined on the leaves of weighted rooted trees (cf. [9, 10, 18]). It is known (see [18]) that for every metric space, there is an HST with stretch $O(\log \mathfrak{m})$. Let $\mathfrak{T}$ be such an HST for the metric given in our instance. Let the level of an internal node in the tree $\mathfrak{T}$ be the number of edges on the path to the root. Let Δ denote the diameter of the resulting metric space. In our paper, we will assume, without loss of generality, that the diameter of the original metric space is $\Delta = 1$. (Indeed, if a pair of points is at distance larger than 1, then in the Facility Location problem we will never allocate one of them to another, since it is always cheaper to pay 1 to open a new facility; therefore, we can treat any distance larger than 1 as equal to 1.) We also modify short distances in the metric, that is, for any pair of points in the metric we assume their distance is at least $1/\mathfrak{c}$. Note that as $\mathfrak{n}_{\mathtt{act}}/\mathfrak{c}$ is a lower bound on the cost of the optimum solution such modification of the metric increases the cost of the optimum solution at most by a constant factor. Then using the framework of HSTs, we will assume that the metric in the instance of capacitated online Facility Location we are solving is a shortest paths metric in a tree $\mathfrak{T}$ of depth $\mathfrak{h} = \log \mathfrak{c}$, satisfying the following conditions:

- any edge connecting vertices of depth i and $i+1$ is of length 2^{-i},
- the set of potential clients are leaves of $\mathfrak{T}$,
- all leaves of $\mathfrak{T}$ are at the same depth $\mathfrak{h}$.

▶ **Definition 4.2.** For two leaves u, v of $\mathfrak{T}$, define $\mathsf{dist\_log}(u, v) = \lfloor -\log \mathsf{dist}_{\mathfrak{T}}(u, v) \rfloor$.

Less formally, $\mathsf{dist\_log}(u, v)$ is the depth of the lowest common ancestor of u and v in $\mathfrak{T}$, which follows from the assumption that the weights of edges of $\mathfrak{T}$ are powers of two depending on their depth. From the triangle inequality we have the following property.

▶ **Claim 4.3.** For $u, v, w \in \mathcal{V}$ we have $\mathsf{dist\_log}(u, w) \geq \min\{\mathsf{dist\_log}(u, v), \mathsf{dist\_log}(v, w)\}$.

4.2 Algorithm for fully dynamic capacitated facility location in HSTs

In this section, we present an algorithm for fully dynamic capacitated Facility Location in a hierarchically well-separated tree $\mathfrak{T}$, Algorithm 2. We will analyze its performance in the next section.

We will assume the input metric space is the shortest path metric in a hierarchically well-separated tree $\mathfrak{T}$, with all input points coming from the leaves of $\mathfrak{T}$, as defined in the previous section. In the algorithm below, when a new facility is opened at a point v, then its capacity $\mathfrak{c}$ is evenly split into $\mathfrak{h}$ parts, denoted as functions $cap_i(v)$ for $0 \leq i < \mathfrak{h}$. We will design the algorithm so that the capacity $cap_i(v)$ will be used solely by clients u such that $\mathsf{dist\_log}(u, v) = i$, that is, by the clients such that the lowest common ancestor of u and v in $\mathfrak{T}$ is of depth i. Apart from that constraint, the algorithm mimics Algorithm 1 from Section 2.1.

Algorithm 2:

insert v:
- Call **connect**(v).

delete v:
- For all clients of v (in arbitrary order) call **connect**(u).

connect v:
- If $\max_v$ is undefined, then set $\max_v = 0$.
- Let u be the closest point to v such that $cap_i(u) > 0$, where $i = \mathsf{dist\_log}(u, v)$.
- If such u does not exist, then **open**(v) and **exit**.
- $p = \min\{1, \mathsf{dist}_{\mathfrak{T}}(u, v) + \frac{12\ \mathfrak{h} \ln \mathfrak{n}}{\mathfrak{c}}\}$.
- If $p \leq 2 \max_v$, then connect v to u and decrease $cap_i(u)$ by one.
- Otherwise: (i) set $\max_v = p$, (ii) with probability p **open**(v), and with probability $1 - p$ connect v to u and decrease $cap_i(u)$ by one.

open v:
- Open a facility at v.
- For each $0 \leq i < \mathfrak{h}$ set $cap_i(v) = \lfloor \mathfrak{c}/\mathfrak{h} \rfloor$.

In the full version of this paper we prove the following main result for the capacitated case of Facility Location with deletion.

▶ **Theorem 4.4.** *There is an $O(\log \mathfrak{m} + \log \mathfrak{c} \log \mathfrak{n})$-competitive online algorithm for capacitated* Facility Location *with deletions.*

5 Conclusions

In this paper we present the first thorough study of natural variants of the online Facility Location problem, when the clients and facilities are not only allowed to arrive to the system, but they can also depart from the system at any moment. In this fully-dynamic online problem, we study two fundamental settings: uncapacitated and capacitated Facility Location for uniform facility costs. For uncapacitated Facility Location, we design an extension of the classical insertion-only randomized algorithm for the Facility Location problem due to Meyerson [40], and show that it achieves an asymptotically optimal competitive ratio of $O(\log \mathfrak{n}_{\mathfrak{act}} / \log \log \mathfrak{n}_{\mathfrak{act}})$ (Theorem 2.2). The capacitated Facility Location is more complex, and here we first show (Observation 3.1) that if no deletions are allowed, then one can achieve an asymptotically optimal competitive ratio of $O(\log \mathfrak{n} / \log \log \mathfrak{n})$, the same bound as it is known for the uncapacitated variant. When deletions are allowed, the task is more challenging, but we still are able to incorporate the framework of hierarchically well-separated trees to obtain an online $O(\log \mathfrak{m} + \log \mathfrak{c} \log \mathfrak{n})$-competitive algorithm for the capacitated Facility Location problem with deletions (Theorem 4.4).

Our work demonstrates that despite the fact that the online Facility Location problem with deletion is clearly more complex than the classical online problem with insertions only, the most natural variants of these problems, for both the uncapacitated and the capacitated model for uniform facility costs, have very efficient online algorithms that achieve competitive ratios matching or almost matching those of the insertion only variants of the problem. It is an interesting open problem whether one can improve the competitive ratio for the capacitated case with deletions, in particular whether it is possible to remove the dependence on $\mathfrak{m}$.

References

1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2012)*, pages 459–467. Society for Industrial and Applied Mathematics, 2012.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In *Proceedings of the 17th International Workshop on Approximation, Randomization, and Combinatorial Optimization (RANDOM'2013)*, pages 1–10. Springer Verlag, Berlin, Heidelberg, 2013.

3 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph (Seffi) Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, October 2006. URL: `http://doi.acm.org/10.1145/1198513.1198522`, `doi:10.1145/1198513.1198522`.

4 Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic facility location via exponential clocks. *ACM Transactions on Algorithms*, 13(2):21:1–21:20, 2017. URL: `http://doi.acm.org/10.1145/2928272`, `doi:10.1145/2928272`.

5 Hyung-Chan An, Mohit Singh, and Ola Svensson. LP-based algorithms for capacitated facility location. *SIAM Journal on Computing*, 46(1):272–306, 2017. URL: `https://doi.org/10.1137/151002320`, `doi:10.1137/151002320`.

6 Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Information and Computation*, 194(2):175–202, 2004. URL: `http://www.sciencedirect.com/science/article/pii/S0890540104001117`, `doi:http://dx.doi.org/10.1016/j.ic.2004.06.002`.

7 Aris Anagnostopoulos, Fabrizio Grandoni, Stefano Leonardi, and Piotr Sankowski. Online network design with outliers. *Algorithmica*, 76(1):88–109, 2016. URL: `https://doi.org/10.1007/s00453-015-0021-y`, `doi:10.1007/s00453-015-0021-y`.

8 Mihai Badoiu, Artur Czumaj, Piotr Indyk, and Christian Sohler. Facility location in sublinear time. In *Proceedings of the 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP'2005)*, volume 3580 of *Lecture Notes in Computer Science*, pages 866–877. Springer Verlag, Berlin, Heidelberg, 2005. `doi:10.1007/11523468_70`.

9 Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science, (FOCS'1996)*, pages 184–193. IEEE Computer Society, 1996. URL: `http://dx.doi.org/10.1109/SFCS.1996.548477`, `doi:10.1109/SFCS.1996.548477`.

10 Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'1998)*, pages 161–168. ACM Press, 1998. URL: `http://doi.acm.org/10.1145/276698.276725`, `doi:10.1145/276698.276725`.

11 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

12 Jaroslaw Byrka and Karen Aardal. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010. URL: `http://dx.doi.org/10.1137/070708901`, `doi:10.1137/070708901`.

13 Graham Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *Proceedings of the 24th ACM Symposium on Principles of Database Systems (PODS'2005)*, pages 271–282. ACM, 2005. URL: `http://doi.acm.org/10.1145/1065167.1065201`, `doi:10.1145/1065167.1065201`.

14 Gérard Cornuéjols, George L. Nemhauser, and Lairemce A. Wolsey. The uncapacitated facility location problem. In Pitu B. Mirchandani and Richard L. Francis, editors, *Discrete Location Theory*, pages 119–171. John Wiley and Son, Inc., New York, 1990.

15 Artur Czumaj, Christiane Lammersen, Morteza Monemizadeh, and Christian Sohler. $(1 + \varepsilon)$-approximation for facility location in data streams. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2013)*, pages 1710–1728. Society for Industrial and Applied Mathematics, 2013. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611973105.123, arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9781611973105.123, doi:10.1137/1.9781611973105.123.

16 Wenqiang Dai and Xianju Zeng. Incremental facility location problem and its competitive algorithms. *Journal of Combinatorial Optimization*, 20(3):307–320, 2010. URL: http://dx.doi.org/10.1007/s10878-009-9219-8, doi:10.1007/s10878-009-9219-8.

17 Gabriella Divéki and Csanád Imreh. Online facility location with facility movements. *Central European Journal of Operations Research*, 19(2):191–200, June 2011. URL: http://dx.doi.org/10.1007/s10100-010-0153-8, doi:10.1007/s10100-010-0153-8.

18 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, November 2004. URL: http://dx.doi.org/10.1016/j.jcss.2004.04.011, doi:10.1016/j.jcss.2004.04.011.

19 Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

20 Dimitris Fotakis. Incremental algorithms for facility location and k-median. *Theoretical Computer Science*, 361(2-3):275–313, 2006. URL: http://www.sciencedirect.com/science/article/pii/S030439750600329X, doi:http://dx.doi.org/10.1016/j.tcs.2006.05.015.

21 Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *Journal of Discrete Algorithms*, 5(1):141–148, 2007. URL: http://www.sciencedirect.com/science/article/pii/S1570866706000268, doi:http://dx.doi.org/10.1016/j.jda.2006.03.001.

22 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008. URL: http://dx.doi.org/10.1007/s00453-007-9049-y, doi:10.1007/s00453-007-9049-y.

23 Dimitris Fotakis. Online and incremental algorithms for facility location. *SIGACT News*, 42(1):97–131, 2011.

24 Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC'2005)*, pages 209–217. ACM Press, 2005. URL: http://doi.acm.org/10.1145/1060590.1060622, doi:10.1145/1060590.1060622.

25 Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2008)*, pages 942–951. Society for Industrial and Applied Mathematics, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347185.

26 Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. *SIAM Journal on Computing*, 42(3):808–830, 2013. URL: http://dx.doi.org/10.1137/100802888, arXiv:http://dx.doi.org/10.1137/100802888, doi:10.1137/100802888.

27 Anupam Gupta and Amit Kumar. Online Steiner tree with deletions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA'2014)*, pages 455–

467. Society for Industrial and Applied Mathematics, 2014. URL: http://dx.doi.org/10.1137/1.9781611973402.34, doi:10.1137/1.9781611973402.34.

28 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'2004)*, pages 373–380. ACM Press, 2004. URL: http://doi.acm.org/10.1145/1007352.1007413, doi:10.1145/1007352.1007413.

29 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

30 Gene Kan. Chapter 8: Gnutella. In Andy Oram, editor, *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*. O'Reilly & Associates, 2001.

31 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. *SIAM Journal on Computing*, 46(1):456–477, 2017. URL: https://doi.org/10.1137/141002281, doi:10.1137/141002281.

32 Michael Kapralov and David P. Woodruff. Spanners and sparsifiers in dynamic streams. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC'2014)*, pages 272–281. ACM Press, 2014.

33 Peter Kling, Friedhelm Meyer auf der Heide, and Peter Pietrzyk. An algorithm for online facility leasing. In *Proceedings of the 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO'2012)*, pages 61–72. Springer Verlag, Berlin, Heidelberg, 2012. URL: http://dx.doi.org/10.1007/978-3-642-31104-8_6, doi:10.1007/978-3-642-31104-8_6.

34 John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. OceanStore: An architecture for global-scale persistent storage. *SIGPLAN Notices*, 35(11):190–201, November 2000. URL: http://doi.acm.org/10.1145/356989.357007, doi:10.1145/356989.357007.

35 Christiane Lammersen and Christian Sohler. Facility location in dynamic geometric data streams. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA'2008)*, pages 660–671. Springer Verlag, Berlin, Heidelberg, 2008. URL: http://dx.doi.org/10.1007/978-3-540-87744-8_55, doi:10.1007/978-3-540-87744-8_55.

36 Jakub Łącki, Jakub Oćwieja, Marcin Pilipczuk, Piotr Sankowski, and Anna Zych. The power of dynamic distance oracles: Efficient dynamic algorithms for the Steiner tree. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC'2015)*, pages 11–20. ACM Press, 2015. URL: http://doi.acm.org/10.1145/1060590.1060622, doi:10.1145/2746539.2746615.

37 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. URL: http://dx.doi.org/10.1016/j.ic.2012.01.007, doi:10.1016/j.ic.2012.01.007.

38 Andrew McGregor. Graph stream algorithms: A survey. *SIGMOD Record*, 43(1):9–20, 2014. URL: http://doi.acm.org/10.1145/2627692.2627694, doi:10.1145/2627692.2627694.

39 Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM Journal on Computing*, 32(3):816–832, 2003. URL: http://www.siam.org/journals/sicomp/32-3/38344.html, doi:10.1137/S0097539701383443.

40 Adam Meyerson. Online facility location. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS'2001)*, pages 426–431. IEEE Computer Society, 2001. URL: http://dl.acm.org/citation.cfm?id=874063.875567.

41 Chandrashekhar Nagarajan and David P. Williamson. Offline and online facility leasing. *Discrete Optimization*, 10(4):361–370, 2013. URL: https://doi.org/10.1016/j.disopt.2013.10.001, doi:10.1016/j.disopt.2013.10.001.

42 Daniel J. Rosenkrantz, Giri K. Tayi, and S.S. Ravi. Obtaining online approximation algorithms for facility dispersion from offline algorithms. *Networks*, 47(4):206–217, 2006. URL: http://dx.doi.org/10.1002/net.20109, doi:10.1002/net.20109.

43 Mário César San Felice, David P. Williamson, and Orlando Lee. A randomized $O(\log n)$-competitive algorithm for the online connected facility location problem. *Algorithmica*, 76(4):1139–1157, 2016. URL: https://doi.org/10.1007/s00453-016-0115-1, doi:10.1007/s00453-016-0115-1.

44 Rajesh Sharma and Anwitaman Datta. Supernova: Super-peers based architecture for decentralized online social networks. In *Proceedings of the 4th Fourth International Conference on Communication Systems and Networks, (COMSNETS'2012)*, pages 1–10. IEEE, 2012. URL: https://doi.org/10.1109/COMSNETS.2012.6151349, doi:10.1109/COMSNETS.2012.6151349.

45 David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'1997)*, pages 265–274. ACM Press, 1997. URL: http://doi.acm.org/10.1145/258533.258600, doi:10.1145/258533.258600.

46 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.

47 Jouni Smed, Timo Kaukoranta, and Harri Hakonen. Networking and multiplayer computer games - the story so far. *International Journal of Intelligent Games & Simulation*, 2(2):101–110, 2003.

48 Seeun Umboh. Online network design algorithms via hierarchical decompositions. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'2015)*, pages 1373–1387. Society for Industrial and Applied Mathematics, 2015. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611973730.91, arXiv:http://epubs.siam.org/doi/pdf/10.1137/1.9781611973730.91, doi:10.1137/1.9781611973730.91.

49 Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'2003)*, pages 49–60. IEEE Computer Society, 2003.