Online and Incremental Algorithms for Facility Location

Dimitris Fotakis

School of Electrical and Computer Engineering
National Technical University of Athens, 15780 Athens, Greece.
Email: fotakis@cs.ntua.gr

Abstract. In the online and incremental variants of Facility Location, the demands arrive one-by-one and must be assigned to an open facility upon arrival, without any knowledge about future demands. In the online variant, the decisions of opening a facility at a particular location and of assigning a demand to some facility are irrevocable. In the incremental variant, the algorithm can also merge existing facilities (and the corresponding demand clusters) with each other, and only the decision of assigning some demands to the same facility is irrevocable. The goal is to maintain, either online or incrementally, a set of facilities and an assignment of the demands to them that minimize the total facility opening cost plus the assignment cost for all demands.

In this survey, we discuss the previous work on online and incremental algorithms for Facility Location. In addition to the main results, namely that the competitive ratio for the online variant is $\Theta(\frac{\log n}{\log \log n})$, where n is the number of demands, and that the competitive ratio for the incremental variant is $\mathrm{O}(1)$, we discuss all known online and incremental Facility Location algorithms, sketch the intuition behind them and the main ideas of their competitive analysis, and discuss some applications.

1 Introduction

Facility Location is a classical problem that has been widely studied in both the operations research and the computer science literature. In this survey, we consider the simplest and most popular variant of Facility Location, namely the *metric uncapacitated Facility Location* problem. The input consists of a metric space, an opening cost for each potential facility location, and a (multi)set of demand locations. The goal is to find a set of facility locations that minimize the opening cost for all facilities plus the assignment cost for all demands, where a demand's assignment cost is the distance of its location to the nearest facility.

Facility Location provides a simple and natural model for industrial planning, network design, content distribution in networks, and data clustering, and has been the subject of a large volume of research in combinatorial optimization and operations research (see e.g., [36, 14]), and more recently, in computer science, especially from the point of view of approximation algorithms (see e.g., [39]). Hence, the approximability of the Facility Location problem is practically fully understood. The best known polynomial-time algorithm is due to Byrka and Aardal [7] and achieves an approximation ratio of 1.5, while no polynomial-time algorithm can achieve an approximation ratio less than 1.463 unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log \log n})$ [22]. Virtually every major technique for the design and analysis of approximation algorithms has been applied to Facility Location. There are constant factor approximation algorithms based on Linear Programming rounding (see e.g., [40, 41]), local search (see e.g., [9, 5]), and the primal-dual method (see e.g., [27]). Moreover, there are greedy algorithms whose analysis is based on Linear Programming duality and the method of dual-fitting [26], and algorithms that combine several of the approaches above (see e.g., [31, 7]).

Online and Incremental Facility Location. The approximation algorithms above require that the set of demand locations is fully known in advance, an assumption questionable in many practical applications. One can think of many natural settings where the demand locations are not known in advance,

and a solution must be maintained incrementally using limited information about future demands. For example, telecommunication networks are usually upgraded incrementally to accommodate new communication demands, because it is expensive, and often infeasible, to reconfigure the complete network every time a few new demands arise.

Motivated mostly by applications to network design, Meyerson [33] introduced the online variant. In *Online Facility Location*, the demands arrive one-by-one and must be irrevocably assigned to an open facility upon arrival. The goal is to maintain a set of facilities and an assignment of the demands to them that minimize the total facility opening cost plus the assignment cost for all demands.

In Online Facility Location, the decisions (and the corresponding costs) of opening a facility at a particular location and of assigning a demand to some facility are irrevocable. This restriction nicely fits network design applications, where once a network device or a physical link is installed somewhere in the network, relocating it is very expensive, and often infeasible. Thus, any decision about the network configuration should be regarded as an irrevocable one.

Nevertheless, the online variant does not really fit applications of Facility Location related to data clustering, where merging clusters is a common practice. Thus, for clustering applications, the decisions of opening a new cluster and of assigning a new demand to a particular cluster center should not be regarded as irrevocable. On the other hand, a major concern in most modern clustering applications is that once formed, clusters should not be broken up. This restriction reflects not only the fact that a hierarchical clustering is often required, but also computational considerations, since splitting existing clusters is usually computationally expensive.

Motivated by clustering applications where the demand sequence is not known in advance and a hierarchical clustering must be maintained incrementally and efficiently, Charikar, Chekuri, Feder, and Motwani [8] introduced the framework of incremental algorithms for optimization problems related to data clustering, such as k-Center, Sum k-Radius, and k-Median. An incremental algorithm processes the demands one-by-one, as they arrive, can create new clusters, and can merge existing clusters at any point in time. However, an incremental algorithm is not allowed to split any existing clusters, and thus the decision of placing some demands in the same cluster is an irrevocable one.

The definition of *Incremental Facility Location*, first studied in [16], follows directly from the framework of incremental algorithms introduced in [8]. In Incremental Facility Location, the demands arrive one-by-one and must be assigned to an open facility upon arrival. At any point in time, the algorithm can also merge a facility with another one by closing the first facility and re-assigning all the demands assigned to it to the second facility. The goal is to maintain incrementally a set of facilities and an assignment of the demands to them that minimize the total facility opening cost plus the assignment cost for all demands.

Aim and Roadmap. In this article, we survey a fair volume of research work on online and incremental algorithms for Facility Location appeared after the introduction of Online Facility Location in [33]. In addition to the main results, we discuss the main ideas behind several online and incremental algorithms and their competitive analysis. In the presentation, we aim to keep a balance between completeness and accuracy, on the one hand, and simplicity and intuition, on the other.

We start, in Section 2, with a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the competitive ratio of any online algorithm for Facility Location [19], where n denotes the number of demands. The lower bound holds for randomized algorithms and for very simple metric spaces, such as the real line.

In Section 3, we present the main ideas behind all known online algorithms for Facility Location with a (near-)optimal competitive ratio. After a brief discussion of the main principles on which the online algorithms are based, we present, in Section 3.1, Meyerson's elegant randomized algorithm [33]. In addition to achieving an asymptotically optimal competitive ratio of $\Theta(\frac{\log n}{\log \log n})$, Meyerson's

algorithm has the remarkable property that it is constant-competitive if the demands arrive in random order.

Next, we proceed to discuss deterministic online algorithms. We start, in Section 3.2, with the primal-dual algorithm of [18], which may be the most intuitive and versatile deterministic algorithm for Online Facility Location. We give a primal-dual interpretation of the algorithm and show that its competitive ratio is $\mathrm{O}(\log n)$. In Section 3.3, we discuss the deterministic algorithm of [19] which achieves an asymptotically optimal competitive ratio. At the conceptual level, the algorithm can be regarded as a delicate derandomization of Meyerson's algorithm. However, the analysis is more complicated, and sheds lights to some deeper issues on the use of locality by online and incremental algorithms. In Section 3.4, we present the simple deterministic algorithm of Anagnostopoulos, Bent, Upfal, and Van Hentenryck [3]. The algorithm applies to the Euclidean plane (more generally, to metric spaces of a small constant dimension) and achieves a competitive ratio of $\mathrm{O}(\log n)$. The algorithm of [3] is remarkably simple to state and implement, faster, and much more space efficient than the algorithm of [19].

In the next two sections, we discuss incremental algorithms for Facility Location. Rather surprisingly, [16] proved that the incremental variant allows for a constant competitive ratio, a result discussed in Section 4. The incremental algorithm of [16] is deterministic, and employs an elegant facility merge rule based on distance considerations. To establish a constant competitive ratio, the analysis shows that in addition to decreasing the algorithm's facility cost, the merge rule succeeds in dramatically decreasing the algorithm's assignment cost.

In Section 5, we consider a relaxed variant of Incremental Facility Location, where the irrevocable decisions only concern any increase in the algorithm's facility cost. In Section 5.1, we present the randomized memoryless algorithm of [17], which maintains a $\mathrm{O}(1)$-competitive facility configuration and keeps in memory only the locations of its facilities and some additional information of constant size per facility. In Section 5.2, we briefly discuss the algorithm of Divéki and Imreh [13], which moves its facilities around and achieves a constant competitive ratio.

In Section 6, we show how one can turn any online or incremental algorithm for Facility Location into an incremental or a streaming algorithm for the closely related problem of k-Median with similar performance characteristics. Thus, the incremental algorithm of [19] leads to an incremental algorithm for k-Median which uses $\mathrm{O}(k)$ medians and achieves a $\mathrm{O}(1)$-competitive ratio, and Meyerson's online algorithm leads to a streaming algorithm for k-Median on general metric spaces with the best known performance characteristics [10].

We conclude, in Section 7, with a brief discussion of some work on approximation and online algorithms for Facility Location that either draws ideas and techniques from or is related to the work on online and incremental algorithms discussed in Sections 3 to 6.

1.1 Problem Definitions, Notation, and Preliminaries

Throughout this article, we consider a metric space (M, d), where M is the set of locations (or points) and $d : M \times M \mapsto \mathbb{N}_+$ is the distance function, which is non-negative, symmetric and satisfies the triangle inequality. For a point $u \in M$ and a subset of points $M' \subseteq M$, we let $d(M', u) \equiv \min_{v \in M'}\{d(v, u)\}$ denote u's distance to the nearest point in M'. We use the convention that $d(\emptyset, u) \equiv \infty$. We let $\mathrm{Ball}(u, r) \equiv \{v \in M : d(u, v) \leq r\}$ denote the set of points within a distance of at most r to u. For every x, y, we let $(x - y)_+ \equiv \max\{x - y, 0\}$.

Facility Location and k-Median. In (metric uncapacitated) Facility Location, the input consists of a metric space (M, d), a facility opening cost f_z for each $z \in M$, and a (multi)set $\{u_1, \ldots, u_n\}$ of

demands in M. The goal is to find a facility configuration $F \subseteq M$ that minimizes

$$\sum_{z \in F} f_z + \sum_{i=1}^{n} d(F, u_i)$$

We highlight that we only consider unit demands and allow multiple demands to be located at the same point. We usually distinguish between the special case of *uniform* facility costs, where the facility opening cost, denoted f, is the same for all points, and the general case of *non-uniform* facility costs, where the facility opening cost f_z depends on the location $z \in M$.

In the closely related problem of k-Median, there is an upper bound of k on the number of facilities, instead of an opening cost for each facility. The input consists of a metric space (M, d) and a (multi)set $\{u_1, \ldots, u_n\}$ of demand locations in M, and the goal is to find a configuration $F \subseteq M$ of at most k facilities (or medians) that minimizes $\sum_{i=1}^{n} d(F, u_i)$.

Online Facility Location. In the online setting, the demand locations arrive one-by-one and must be assigned to an open facility upon arrival. For the special case of uniform facility costs, the input consists of the facility opening cost f and a sequence $u_1, \ldots, u_n$ of (not necessarily distinct) demand locations in an underlying metric space (M, d). The online algorithm maintains its facility configuration in response to the demand sequence $u_1, \ldots, u_n$. Throughout this article, we let F_i denote the algorithm's facility configuration just after demand u_i is processed, with $F_0 = \emptyset$. Since an online algorithm can only open facilities, $F_{i-1} \subseteq F_i$ for any $i \geq 1$.

Upon arrival of a demand u_i, the online algorithm applies a so-called *facility opening rule*, which takes into account the cost f of opening a new facility, the algorithm's current facility configuration F_{i-1}, the location of u_i, and possibly the locations of (some of) the past demands $u_1, \ldots, u_{i-1}$, and determines whether and at which location a new facility should open. If a new facility opens at location w, $F_i = F_{i-1} \cup \{w\}$, and the algorithm's facility cost increases by f. Otherwise, $F_i = F_{i-1}$. Finally, u_i is assigned to the nearest facility in F_i, and the algorithm's assignment cost increases by $d(F_i, u_i)$. Therefore, the algorithm's cost just after demand u_i is processed is:

$$|F_i| f + \sum_{\ell=1}^{i} d(F_\ell, u_\ell)$$

In the general case of non-uniform facility costs, the underlying metric space M along with the facility opening cost f_z for each $z \in M$ are given to the algorithm in advance. The online algorithm maintains its facility configuration $F_0 = \emptyset, F_1, \ldots, F_n$ in response to the demand sequence $u_1, \ldots, u_n$ as above. The only difference is that the facility opening rule should now take the different facility opening costs into account. The algorithm's cost just after demand u_i is processed is:

$$\sum_{z \in F_i} f_z + \sum_{\ell=1}^{i} d(F_\ell, u_\ell)$$

Incremental Facility Location and k-Median. In addition to opening new facilities and assigning new demands to them, an incremental algorithm for Facility Location can merge existing facilities (and the corresponding demand clusters) with each other.

For the incremental variant, we restrict our attention to uniform facility costs. Similarly to the online variant, the input consists of the facility opening cost f and a sequence $u_1, \ldots, u_n$ of demands which arrive online. The incremental algorithm maintains its facility configuration $F_0 = \emptyset, F_1, \ldots, F_n$

and a clustering of the demands processed so far in response to the demand sequence $u_1, \ldots, u_n$. For each facility $z \in F_i$, the algorithm maintains the set $C(z)$ of the demands currently assigned to z (or z's cluster), and possibly some additional information about z. Just after each demand u_i is processed, the union of $C(z)$ over all $z \in F_i$ must be $\{u_1, \ldots, u_i\}$.

When a demand u_i arrives, the incremental algorithm applies a facility opening rule and determines whether u_i should be assigned to an existing facility, or a new facility should open and u_i should be assigned to it. The algorithm also applies a so-called *merge rule* and determines whether some facilities should be merged with each other. If a facility z is merged with a facility z', z is closed, i.e., it is removed from F_i, and the demands currently assigned to z are reassigned to z'. The cost of the incremental algorithm just after demand u_i is processed is:

$$|F_i|f + \sum_{z \in F_i} \sum_{u \in C(z)} d(u, z)$$

The incremental variant[1] of k-Median is defined similarly. The only difference is that there is no opening cost associated with each facility / median, and the number of medians maintained by the algorithm should not exceed k by too much.

Preliminaries. We evaluate the performance of online and incremental algorithms using *competitive analysis* (see e.g., [6]). A deterministic (resp. randomized) algorithm is c-competitive if for any demand sequence, its cost (resp. expected cost) is at most c times the optimal cost for the corresponding offline Facility Location instance, where the demand sequence is fully known in advance.

For the competitive analysis, we fix a sequence of n demands, and compare the algorithm's cost against the cost of a (fixed offline) optimal solution. We let Fac^* denote the facility cost and Asg^* denote the assignment cost of the optimal solution. To avoid confusing an algorithm's facility with a facility of the optimal solution, we use the term *optimal center*, or simply *center*, to refer to the latter.

To sketch the main idea of an algorithm's competitive analysis, we focus on a single optimal center c and the demands assigned to it in the optimal solution. We let $d^*_u = d(u, c)$ denote the optimal assignment cost of a demand u assigned to c, let $\mathrm{Asg}^*(c) = \sum_u d^*_u$ denote the optimal assignment cost for the demands assigned to c, and let δ^* denote the average optimal assignment cost for them.

2 A Lower Bound for Online Algorithms

We start with a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the competitive ratio of any online algorithm for Facility Location [19]. 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 the lower bound construction, we employ a metric space described by a Hierarchically Well-Separated Tree (see e.g., [15]). In particular, for some positive integers h, m, and a positive rational D (all of them depend on n and f), we consider a complete binary tree T of height h such that (i) the distance of the root to its children is D, and (ii) on every path from the root to a leaf, the edge length drops by a factor of m on every level. Thus, the distance of any vertex at level j to its children

[1] The online and incremental variants of Facility Location and k-Median should not be confused with the problem of *Online Median*, which was introduced in [32] and admits a constant competitive ratio (see e.g., [12, Section 9] for a brief discussion of previous work on Online Median). In Online Median, the demand locations are fully known in advance and the number of facilities / medians increases online. The goal is to compute a permutation of the demand locations so that for any integer k, opening facilities at the first k locations yields a good approximate solution to the corresponding k-Median instance. Thus, the setting of Online Median is somehow orthogonal to the setting of Online and Incremental Facility Location, where the facility cost is known in advance and the demands arrive online.

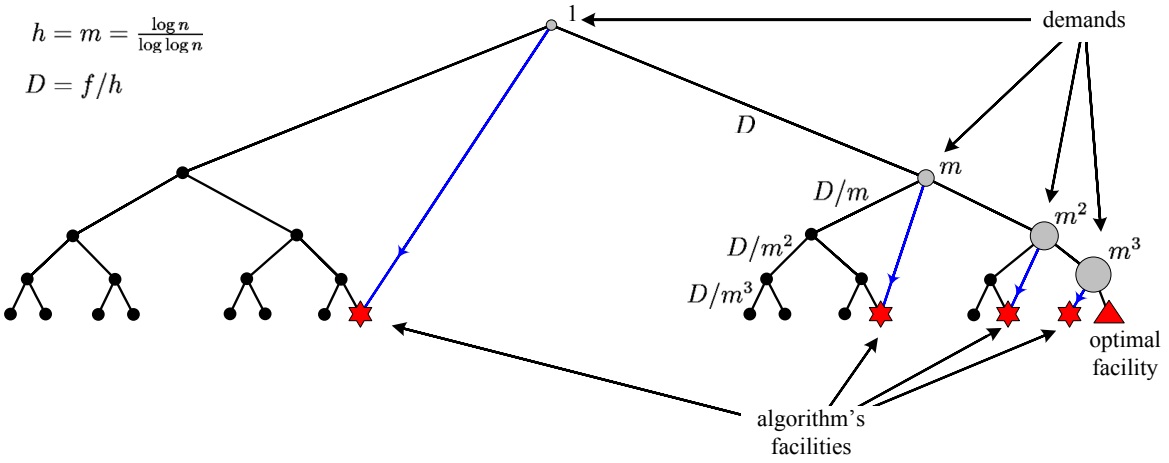

Fig. 1. The metric space and the demand sequence used in the lower bound construction.

is D/m^j. We let T_v be the subtree of T rooted at a vertex v. The key property of T is that the distance of a level-j vertex v to any vertex in T_v is at most $\frac{D}{m^{j-1}(m-1)} \approx D/m^j$, while the distance of v to any vertex not in T_v is at least D/m^{j-1}.

We first show the lower bound for any deterministic algorithm Alg. The demand sequence is divided into $h+1$ phases. Phase 0 consists of a single demand located at the root v_0 of T. After the end of phase j, $0 \le j \le h-1$, the location v_{j+1} of the demands to arrive in phase $j+1$ is selected among the children of v_j. In particular, v_{j+1} is the right child of v_j, if Alg does not have any facilities in the right subtree of T_v, and the left child of v_j, otherwise. The $(j+1)$-th phase consists of m^{j+1} unit demands located at v_{j+1}. Thus, the total number of demands is at most $m^{h+1}/(m-1) \approx m^h$.

If we open a single facility at v_h, the assignment cost for the demands in each of the first h phases is at most $D\frac{m}{m-1}$. Therefore, the optimal cost is at most $f + hD\frac{m}{m-1}$. On the other hand, Alg incurs a cost of at least $\min\{f, mD\}$ for each phase. Intuitively, at to the end of phase $j-1$, Alg knows that there must be a facility in $T_{v_{j-1}}$, but it cannot tell in which of $T_{v_{j-1}}$'s subtrees the facility should be. Thus Alg opens a facility either in both subtrees of $T_{v_{j-1}}$, or in one of them, or in none of them. In any case, Alg's cost for the demands and the facilities in $T_{v_{j-1}}$ but not in T_{v_j} is at least $\min\{f, mD\}$. If Alg does not have a facility in $T_{v_{j-1}}$, its assignment cost for the demands at v_{j-1} is at least $m^{j-1}D/m^{j-2} = mD$. Otherwise, v_j is selected so that (at least) one of Alg's facilities in $T_{v_{j-1}}$ is not included in T_{v_j}. Taking the first h phases into account, Alg's cost for the demands and the facilities not in T_{v_h} is at least $h\min\{f, mD\}$. In addition, Alg incurs a cost of $\min\{f, mD\}$ for the demands located at v_h. Thus, the total cost of Alg is at least $(h+1)\min\{f, mD\}$.

Setting $m = h$ and $D = f/h$, we conclude that Alg's total cost for the demand sequence above is at least $h/2$ times the optimal cost. Since the number of demands, which is roughly m^h, should not exceed n, we use $h = \left\lfloor \frac{\log n}{\log \log n} \right\rfloor$, and obtain a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the competitive ratio of any deterministic online Facility Location algorithm.

We can use Yao's principle (see e.g., [6, Chapter 8]) and extend the lower bound to randomized algorithms. As before, the metric space is given by T and the demand sequence consists of $h+1$ phases, with m^j arriving in each phase j. The only difference is that given v_j, the location v_{j+1} of the demands arriving in the next phase $j+1$ is selected uniformly at random among v_j's children.

Adapting the arguments above and applying Yao's principle, we obtain a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the competitive ratio of any randomized algorithm against the oblivious adversary.

Moreover, the lower bound above holds even if the metric space is the real line. At the conceptual level, if n (and thus m) is large enough, the binary tree T is very close to a line segment. Based on this observation, [19] describes an embedding of T in the line that preserves the main properties of T.

3 Online Algorithms for Facility Location

Next, we discuss the main ideas behind online Facility Location algorithms with a (near-)optimal competitive ratio. For simplicity, we mostly focus on the special case of uniform facility costs, and discuss which algorithms can be generalized to non-uniform facility costs.

Online Facility Location algorithms differ from each other with respect to their facility opening rules. Nevertheless, all known facility opening rules are based on two main principles. The first principle is to maintain a balance between the algorithm's facility and assignment cost. This is typically implemented by allocating (implicitly or explicitly) a (facility opening) *potential* of $d(F_{i-1}, u_i)$ to each demand u_i. The potential of u_i serves as an upper bound both on u_i's assignment cost and on how much u_i can contribute to the opening cost of new facilities closer to it. Thus the total potential allocated to the demands gives an upper bound both on the assignment and on the facility cost of the algorithm.

The second principle is that the facility opening potential accumulated in any (appropriately defined and typically small) area should not exceed (by too much) the cost of opening a facility. This invariant is motivated by the dual constraint of the Linear Programming relaxation for the Facility Location problem, with each demand's potential playing the role of the corresponding dual variable (cf. (DP) in Section 3.2). The facility opening rule ensures that a new facility opens in an area by the moment when the potential accumulated in it exceeds the facility opening cost. When a new facility opens, the potential of the nearby demands decreases accordingly, and the invariant is restored. The main difference between known online algorithms lies in how they maintain this invariant.

3.1 Meyerson's Randomized Algorithm

Meyerson [33] presented an elegant randomized algorithm with an (asymptotically) optimal competitive ratio and a few other interesting properties. Meyerson's algorithm, or $\mathrm{RandOFL}$ in short, employs a remarkably simple and intuitive facility opening rule: each new demand u_i opens a new facility at u_i's location with probability $d(F_{i-1}, u_i)/f$. Then u_i is assigned to the nearest facility available.

If demand u_i opens a new facility, the algorithm incurs a facility cost of f and no assignment cost. Otherwise, the algorithms incurs an assignment cost of $d(F_{i-1}, u_i)$ and no facility cost. Hence $\mathrm{RandOFL}$'s expected assignment and expected facility cost due to u_i both are at most $d(F_{i-1}, u_i)$.

$\mathrm{RandOFL}$ implements a stochastic version of the second principle. Using a simple potential function argument (or expected waiting time techniques), one can show that the expected cost of $\mathrm{RandOFL}$ due to any demand set D by the moment when the first facility in D opens is at most $2f$. This cost is made up of an expected assignment cost of at most f due to the demands in D arriving before the first facility in D opens and a facility cost of f for the first facility in D.

Competitive Analysis. The properties above imply that the competitive ratio of $\mathrm{RandOFL}$ is at most $2(h + m + 3)$, where m, h are any positive integers with $m^h > n$. To sketch the main ideas of the analysis, we focus on a single optimal center c and the demands assigned to it in the optimal solution.

We break down the analysis of $\mathrm{RandOFL}$ into $h + 2$ disjoint phases, numbered $h, h-1, \ldots, 0$ as time goes, according to the distance of the algorithm's facility configuration to c. The j-th phase,

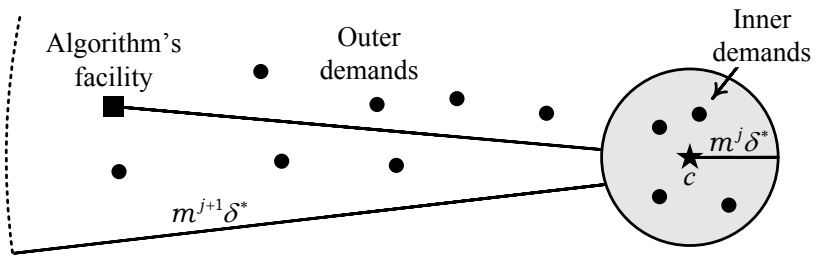

Fig. 2. In the analysis of $\mathrm{RandOFL}$, phase j begins just after a facility within a distance of $m^{j+1}\delta^*$ to c opens. The demands in $\mathrm{Ball}(c, m^j\delta^*)$, i.e. the grey ball around c, are called *inner* and the remaining demands are called *outer*. Phase j ends as soon as a facility within a distance of $m^j\delta^*$ to c opens, or equivalently, as soon as some inner demand opens a new facility.

$j = h, \ldots, 0$, begins just after a facility within a distance of $m^{j+1}\delta^*$ to c opens, and ends as soon as a facility within a distance of $m^j\delta^*$ to c opens. There is also a final phase, which follows phase 0 and never ends. The demands arriving in each phase are classified into *inner* and *outer* demands. A demand u arriving in phase $j \geq 0$ is inner if $d^*_u < m^j\delta^*$ and outer otherwise (see also Fig. 2). Since $m^h\delta^* > n\delta^* \geq \mathrm{Asg}^*(c)$, all demands arriving in (the first) phase h are inner. We also use the convention that all demands arriving in the last phase are outer[2].

Then, bounding the expected cost due to outer demands is not really difficult. By definition, the distance of an outer demand to c is sufficiently large, so large that the algorithm's expected cost due to it can be charged to its optimal assignment cost. More precisely, if a demand u_i arrives in a non-final phase j as an outer demand, we have that

$$d(F_{i-1}, u_i) \leq d(F_{i-1}, c) + d^*_{u_i} \leq (m+1)d^*_{u_i}\,,$$

because by the definition of phase j, $d(F_{i-1}, c) \leq m^{j+1}\delta^* \leq md^*_{u_i}$. Therefore, the expected increase in $\mathrm{RandOFL}$'s cost due to u_i is at most $2(m+1)d^*_{u_i}$. Similarly, the expected increase in $\mathrm{RandOFL}$'s cost due to a demand u_i arriving in the final phase is at most $2(\delta^* + d^*_{u_i})$, since $d(F_{i-1}, c) \leq \delta^*$. Therefore, the algorithm's expected cost due to outer demands is at most $2(m+2)\mathrm{Asg}^*(c)$.

One should be more careful with inner demands, because they are located very close to c. Hence, the optimal solution incurs a negligible assignment cost due to them, while the algorithm's cost can be quite large. Hence any reasonable algorithm should open a new facility in (or, at least, very close to) the area occupied by the inner demands of the current phase, before their assignment cost becomes much larger than f. The definition of phases aims precisely to quantifying this point. In fact, the phases for $\mathrm{RandOFL}$ are defined so that phase j starts when a phase-$(j+1)$ inner demand opens a new facility, and ends when a phase-j inner demand opens a new facility. Since the expected cost due to the inner demands arriving by the moment when the first of them opens a new facility is at most $2f$, $\mathrm{RandOFL}$'s expected cost due to the inner demands arriving in phase j is at most $2f$, and the total expected cost due to inner demands is at most $2(h+1)f$.

Putting everything together, we obtain that the expected cost of $\mathrm{RandOFL}$ is bounded from above by $2(h+1)\mathrm{Fac}^* + 2(m+2)\mathrm{Asg}^*$, which implies a competitive ratio of $2(h+m+3)$. Setting $m = h = \Theta(\frac{\log n}{\log\log n})$, we conclude that $\mathrm{RandOFL}$ is $\Theta(\frac{\log n}{\log\log n})$-competitive.

[2] We should highlight the correspondence between the phases here and the phases in the lower bound construction, in Section 2. If we consider $\mathrm{RandOFL}$ against the demand sequence used in the lower bound construction, phase j starts as soon as a facility at v_{j-1} opens. The demands located at v_j and arriving in phase j are all inner demands. Phase j lasts until the algorithm opens a facility at v_j. During each phase j, $\mathrm{RandOFL}$ incurs an expected assignment cost of at most f for the demands located at v_j and arriving before a facility at v_j opens, and a cost of f for the new facility at v_j.

Random Demand Arrivals. A remarkable property of $\mathrm{RandOFL}$ is that it achieves a constant competitive ratio when the demands arrive in random order [33, Theorem 2.1]. In particular, $\mathrm{RandOFL}$ is 8-competitive if the demands are selected by an oblivious adversary and are presented to the algorithm according to a random permutation of them. At the intuitively level, this implies that for any set of demands, $\mathrm{RandOFL}$ performs very well against most of the demand orderings.

To sketch the analysis of the competitive ratio, we again focus on a single optimal center c and the demands assigned to it in the optimal solution. We now consider a single phase that never ends, and classify as inner the half of the demands closer to c. Hence the distance of an inner demand to c is at most $2\delta^*$. The remaining half of the demands are outer.

As before, the expected cost due to the inner demands arriving by the moment when the first of them opens a facility is at most $2f$. The expected cost due to an inner demand u arriving afterwards is at most $2(2\delta^* + d_u^*)$, because then u arrives, there is a facility within a distance of $2\delta^*$ to c. Using that the inner demands are the half of the demands closer to c, we obtain that the expected cost due to inner demands is at most $2f + 3\mathrm{Asg}^*(c)$ in total. Moreover, this upper bound holds despite their arrival order, an observation that facilitates the analysis of the expected cost due to outer demands.

As for the outer demands, we consider an ordering obtained by randomly injecting them into some fixed ordering of the inner demands. In such an ordering, if an outer demand u arrives before any inner demand, the cost due to u is at most f. Otherwise, let v be the most recent inner demand arriving before u. Since the expected cost due u (resp. v) is at most twice its distance to the nearest facility when u (resp. v) arrives, the expected cost due to u is at most $2d(u, v) \leq 2(d_u^* + d_v^*)$ plus the expected cost due to v. Moreover, v is equally likely to be any inner demand, because the outer demands are randomly injected among the inner demands. Using that there are as many inner demands as outer demands, one can show that the expected cost due to outer demands is at most $f + 2\mathrm{Asg}^*(c)$ plus the expected cost due to inner demands.

Putting everything together, the total expected cost of $\mathrm{RandOFL}$ due to the demands assigned to c in the optimal solution is at most:

$$\overbrace{2f + 3\mathrm{Asg}^*(c)}^{\text{cost for inner demands}} + \overbrace{f + 2\mathrm{Asg}^*(c) + (2f + 3\mathrm{Asg}^*(c))}^{\text{cost for outer demands}} = 5f + 8\mathrm{Asg}^*(c)$$

Taking all optimal centers into account, the expected cost of $\mathrm{RandOFL}$ is at most $5\mathrm{Fac}^* + 8\mathrm{Asg}^*$, which implies an expected competitive ratio of 8.

Non-Uniform Facility Costs. $\mathrm{RandOFL}$ naturally generalizes to non-uniform facility costs. Its competitive ratio is $\Theta(\frac{\log n}{\log \log n})$, when the demands arrive in an adversarial order, and at most 33, when the demands arrive in random order [33, Theorem 3.1].

In case of non-uniform facility costs, the underlying metric space M and the facility opening cost f_z for each location $z \in M$ are given to the algorithm in advance. For simplicity, we assume that the minimum facility cost is 1. The algorithm first computes the *type* $t_z = \lfloor \log_2 f_z \rfloor$ of each location z (this is equivalent to rounding down the facility costs to the nearest integral power of 2). For each $t \geq 0$, we let $F^{(t)}$ denote the set of locations of type at most t, with $F^{(-1)} = \emptyset$. When a demand u_i arrives, $\mathrm{RandOFL}$ opens a facility at the location of type t closest to u_i with probability $(d(F^{(t-1)} \cup F_{i-1}, u_i) - d(F^{(t)} \cup F_{i-1}, u_i))/2^t$, for each type $t \geq 0$ independently.

Intuitively, each demand u_i is allocated a (facility opening) potential of $d(F_{i-1}, u_i)$, which in turn is distributed to each facility type t according to the distance of u_i to $F^{(t)}$. More precisely, for each type t, we let z_t denote the location in $F^{(t)} \cup F_{i-1}$ closest to u_i (note that $F^{(t-1)} \subseteq F^{(t)}$). Then each z_t receives a portion of u_i's potential equal to $d(z_{t-1}, u_i) - d(z_t, u_i)$. The probability that a new facility at z_t opens is obtained by dividing z_t's portion of u_i's potential by z_t's opening cost, namely 2^t.

In case of adversarial demand arrivals, the analysis proceeds as that for uniform facility costs. The expected increase in $\mathrm{RandOFL}$'s cost due to each demand u_i is again bounded by $2d(F_{i-1}, u_i)$. As before, we focus on the demands assigned to an optimal center c, fix some integers h, m with $m^h > n$, and divide the analysis into $h + 2$ phases. The distance of c to the nearest facility decreases roughly by a factor of m in each phase. We again distinguish between inner and outer demands, and show that the expected increase in the algorithm's cost due to an outer demand u is $\mathrm{O}(md^*_u + \delta^*)$.

The only essential difference has to do with the analysis of the algorithm's cost due to the inner demands arriving in each phase. Similarly to the case of uniform facility costs, phase j starts when a phase-$(j + 1)$ inner demand opens a new facility of type at least t_c and ends when a phase-j inner demand opens a new facility of type at least t_c, where t_c is the facility type of the optimal center c. But now the analysis should account for the facilities of smaller types that inner demands may open. To this end, each phase is subdivided into $t_c + 1$ *stages*. Each phase starts in stage 0 and ends when its final stage t_c is completed. Within a given phase, stage t starts when an inner demand opens a facility of type $t - 1$, and ends when an inner demand opens a facility of type t. The crucial step is to show that most of the potential of each inner demand u arriving in stage t is contributed towards opening new facilities of type at least t. Then, by a potential function argument, one can show that the expected cost due to the inner demands arriving in stage t is at most $\mathrm{O}(2^t)$ plus a constant times their optimal assignment cost. Therefore, the expected cost due to the inner demands arriving in the entire phase phase is at most $\mathrm{O}(f_c)$ plus a constant times their optimal assignment cost.

Putting everything together, we obtain that the expected facility cost and the expected assignment cost of $\mathrm{RandOFL}$ are both at most $\mathrm{O}(h)\mathrm{Fac}^* + \mathrm{O}(m)\mathrm{Asg}^*$. Setting $m = h = \Theta(\frac{\log n}{\log\log n})$, we conclude that the competitive ratio of $\mathrm{RandOFL}$ for non-uniform facility costs is $\Theta(\frac{\log n}{\log\log n})$.

3.2 A Deterministic Primal-Dual Algorithm

We proceed to discuss the primal-dual algorithm of [18], which is quite simple and intuitive. In a nutshell, the algorithm maintains its facility configuration so that at any point in time, the distances of the demands processed so far to the algorithm's facilities comprise a dual feasible solution.

Due to the inherent simplicity of the primal-dual algorithm, or PD-OFL in short, we only discuss the version dealing with non-uniform facility costs. The algorithm is aware of the metric space M and of the facility opening cost f_z for each $z \in M$, and maintains its facility configuration $F_0 = \emptyset, F_1, \ldots, F_n$ in response to the demand sequence $u_1, \ldots, u_n$. When a demand u_i arrives, PD-OFL calculates the (facility opening) potential $p_i(z) = \sum_{\ell=1}^{i}(d(F_{i-1}, u_\ell) - d(z, u_\ell))_+$ of each location $z \in M$, and seeks for the location w maximizing $p_i(w) - f_w$. If $p_i(w) > f_w$, PD-OFL opens a new facility at w. Otherwise, no new facilities open. Finally, u_i is assigned to the nearest facility in F_i.

Intuitively, each demand u_i is allocated a potential of $d(F_{i-1}, u_i)$, which u_i can contribute towards opening new facilities closer to it. In particular, if a new facility at z opens, the potential of each demand u_i becomes $d(F_{i-1} \cup \{z\}, u_i) \leq d(F_{i-1}, u_i)$. Thus, u_i contributes the difference $(d(F_{i-1}, u_i) - d(z, u_i))_+$ in its potential towards opening a new facility at z. The (facility opening) potential $p_i(z)$ of z corresponds to the total decrease in the potential of demands $u_1, \ldots, u_i$ if a new facility at z opens. PD-OFL opens a new facility only if $p_i(z)$ exceeds the cost f_z of opening a new facility at z. Moreover, the location of the new facility is selected so as to maximize the potential surplus $p_i(z) - f_z$. Thus, PD-OFL maintains the invariant that the potential accumulated in any location z never exceeds f_z. Whenever the invariant is violated due to the arrival of a new demand, a new facility opens at the location with the maximum potential surplus, and the invariant is restored.

Primal-Dual Interpretation. The approach above is motivated by Linear Programming duality. The offline version of Facility Location is naturally formulated as the following $0-1$ Integer Program:

$$\begin{array}{lccll} \min & \sum_{z\in M} f_z y_z + \sum_{z\in M}\sum_{i=1}^{n} x_{zi} d(z,u_i) & & & \\ \text{s.t} & \sum_{z\in M} x_{zi} = 1 & & \forall \text{ demand } u_i & \\ & x_{zi} \le y_z & & \forall z\in M,\ \forall \text{ demand } u_i & \text{(IP)} \\ & y_z \in \{0,1\}, x_{zi}\in\{0,1\} & & \forall z\in M,\ \forall \text{ demand } u_i & \end{array}$$

Setting y_z to 1 corresponds to opening a facility at z, and setting x_{zi} to 1 corresponds to assigning demand u_i to facility z. We obtain a Linear Programming relaxation of (IP) by replacing the 0-1 constraints to $y_z \ge 0$ and $x_{zi} \ge 0$ respectively. The dual of the Linear Programming relaxation is:

$$\begin{array}{lccll} & \max \sum_{i=1}^{n} \alpha_i & & & \\ \text{s.t} & \alpha_i \le \beta_{zi} + d(z,u_i) & & \forall z\in M,\ \forall \text{ demand } u_i & \text{(DP}'\text{)} \\ & \sum_{i=1}^{n} \beta_{zi} \le f_z & & \forall z\in M & \\ & \alpha_i \ge 0,\ \beta_{zi}\ge 0 & & \forall z\in M,\ \forall \text{ demand } u_i & \end{array}$$

Observing that $\beta_{zi} \ge (\alpha_i - d(z,u_i))_+$, we obtain the following simplified version of the dual:

$$\begin{array}{lccll} & \max \sum_{i=1}^{n} \alpha_i & & & \\ \text{s.t} & \sum_{i=1}^{n} (\alpha_i - d(z,u_i))_+ \le f_z & & \forall z\in M & \text{(DP)} \\ & \alpha_i \ge 0 & & \forall \text{ demand } u_i & \end{array}$$

PD-OFL simply maintains its facility configuration so that after each demand u_i is processed, the distances $d(F_i,u_\ell)$, $\ell \le i$, comprise a feasible solution to (DP) for demands $u_1,\ldots,u_i$. Whenever a dual constraint is violated due to the arrival of a new demand, a new facility opens at the location corresponding to the most violated dual constraint, and dual feasibility is restored.

Dual Feasibility. We first discuss why opening a new facility at the location corresponding to the most violated dual constraint indeed restores dual feasibility. To this end, let us inductively assume that just after demand u_{i-1} is processed, the distances $d(F_{i-1},u_\ell)$, $\ell \le i-1$, comprise a feasible solution to (DP), and when demand u_i arrives, some dual constraint is violated.

More precisely, we let $p'_{i-1}(z) = \sum_{\ell=1}^{i-1} (d(F_{i-1},u_\ell) - d(z,u_\ell))_+$ denote the potential of a location z just after demand u_{i-1} is processed, and assume that $p'_{i-1}(z) \le f_z$ for all locations z. Let w be the location at which the new facility opens when u_i arrives. For any location z not in $\mathrm{Ball}(u_i, d(w,u_i))$, u_i does not contribute to z's potential after w opens. Thus, z's potential after w opens is at most $p'_{i-1}(z) \le f_z$. If $d(z,u_i) \le d(w,u_i)$, we recall that w maximizes the potential surplus $p_i(w) - f_w$, and that for any z in $\mathrm{Ball}(u_i, d(w,u_i))$, $p_i(z) = p'_{i-1}(z) + d(F_{i-1},u_i) - d(z,u_i)$ (see also Fig 3.a). Subtracting $d(F_{i-1},u_i) - d(w,u_i)$, namely the contribution of u_i to w's potential, from both sides of $p_i(w) - f_w \ge p_i(z) - f_z$, we obtain that (see also Fig 3.b):

$$p'_{i-1}(w) - f_w \ge p'_{i-1}(z) - f_z + d(w,u_i) - d(z,u_i)$$

In this inequality, the left-hand side is non-positive, since $p'_{i-1}(w) \le f_w$ before u_i arrives. The right-hand side is no less than $p'_i(z)$, namely z's potential after w opens, minus f_z. Therefore, $p'_i(z) \le f_z$. So w's opening indeed restores the dual feasibility of the distances $d(F_i,u_\ell)$, $\ell \le i$.

Competitive Analysis. We proceed to show that the competitive ratio of PD-OFL is at most $4H_n - 2$, where H_n denotes the n-harmonic number. We first observe that the cost of PD-OFL due to any demand u_i is at most $2\mathrm{cost}(u_i)$, where $\mathrm{cost}(u_i) = \min\{d(F_{i-1},u_i), \min_z\{f_z - p'_{i-1}(z) + d(z,u_i)\}\}$.

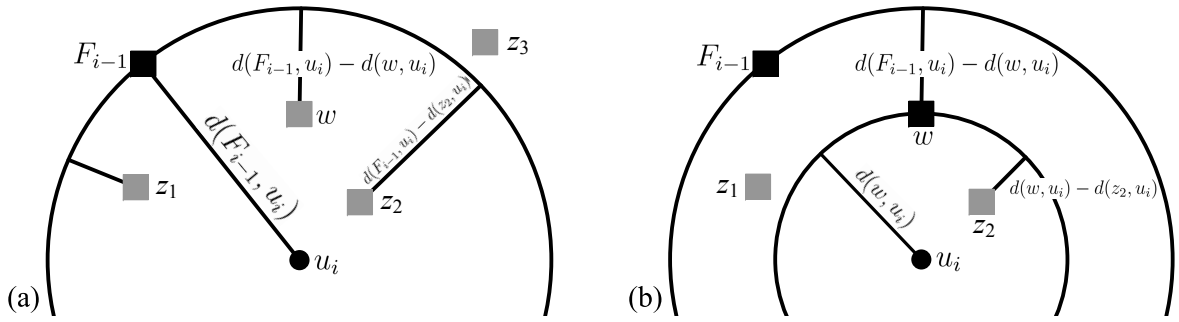

Fig. 3. (a) When a demand u_i arrives, it is allocated a potential of $d(F_{i-1}, u_i)$. Demand u_i contributes an amount of $d(F_{i-1}, u_i) - d(z, u_i)$ to the potential of every location z in $\mathrm{Ball}(u_i, d(F_{i-1}, u_i))$ (e.g., w, z_1, and z_2), and does not contribute to the potential of any location z not in $\mathrm{Ball}(u_i, d(F_{i-1}, u_i))$ (e.g., z_3). (b) After the new facility w opens, the potential of u_i decreases to $d(w, u_i)$, and u_i stops contributing to the potential of any location z not in $\mathrm{Ball}(u_i, d(w, u_i))$ (e.g., z_1). Furthermore, the contribution of u_i to the potential of each location z in $\mathrm{Ball}(u_i, d(w, u_i))$ decreases by $d(F_{i-1}, u_i) - d(w, u_i)$.

If a demand u_i does not open a new facility, PD-OFL incurs an assignment cost of $d(F_{i-1}, u_i)$ due to u_i. Otherwise, PD-OFL's cost is equal to u_i's contribution to the cost of the new facility w, namely $f_w - p'_{i-1}(w)$, plus u_i's assignment cost, namely $d(w, u_i)$. In both cases, u_i's assignment cost must be doubled to account for u_i's (facility opening) potential, which u_i can spend towards opening new facilities closer to it. Since PD-OFL greedily chooses the minimum cost action for any demand, the total cost due to u_i is at most $2\mathrm{cost}(u_i)$. More precisely, a demand u_i does not open a new facility if and only if for all locations z,

$$p_i(z) - f_z = p'_{i-1}(z) - f_z + (d(F_{i-1}, u_i) - d(z, u_i))_+ \leq 0\,, \tag{1}$$

which is equivalent to $d(F_{i-1}, u_i) \leq \min_z\{f_z - p'_{i-1}(z) + d(z, u_i)\}$. Then, the total cost of PD-OFL due to u_i is at most $2d(F_{i-1}, u_i)$. On the other hand, if $d(F_{i-1}, u_i) > \min_z\{f_z - p'_{i-1}(z) + d(z, u_i)\}$, u_i opens a new facility at the location w that maximizes the left-hand side of (1), or equivalently minimizes $f_w - p'_{i-1}(w) + d(w, u_i)$. Then, the total cost of PD-OFL due to u_i is at most $f_w - p'_{i-1}(w) + 2d(w, u_i)$. In both cases, the total cost of PD-OFL due to u_i is at most $2\mathrm{cost}(u_i)$.

The intuition behind the logarithmic competitive ratio is that the invariant maintained by the algorithm implies that $\mathrm{cost}(u_i)$ is bounded from above by a fraction of the optimal cost that decreases harmonically with i. To formalize this intuition, we consider a single optimal center c and assume that all demands $u_1, \ldots, u_n$ are assigned to c in the optimal solution. We first observe that by the definition of $\mathrm{cost}(u_i)$:

$$\begin{aligned} f_c &\geq \mathrm{cost}(u_i) - d(c, u_i) + p'_{i-1}(c) \\ &\geq \mathrm{cost}(u_i) - d(c, u_i) + \sum_{\ell=1}^{i-1} (d(F_{i-1}, u_\ell) - d(c, u_\ell)) \end{aligned}$$

Using that $d(F_{i-1}, u_\ell) \geq d(F_{i-1}, u_i) - d(c, u_i) - d(c, u_\ell)$, that $d(F_{i-1}, u_i) \geq \mathrm{cost}(u_i)$, and that $d(c, u_\ell) = d^*_{u_\ell}$ is the optimal assignment cost of each demand u_ℓ, we obtain that:

$$f_c \geq i(\mathrm{cost}(u_i) - d^*_{u_i}) - 2\sum_{\ell=1}^{i-1} d^*_{u_\ell}\,,$$

which can be rewritten as:

$$\text{cost}(u_i) \leq \frac{1}{i} f_c + d^*_{u_i} + \frac{2}{i} \sum_{\ell=1}^{i-1} d^*_{u_\ell}$$

Summing up over all demands u_i assigned to c in the optimal solution, we obtain that:

$$\sum_{i=1}^{n} \text{cost}(u_i) \leq H_n f_c + (2H_n - 1)\text{Asg}^*(c)$$

Multiplying by 2, we conclude that the competitive ratio of PD-OFL is at most $4H_n - 2$.

Nagarajan and Williamson [35] suggested that PD-OFL is similar to the 1.61-approximation greedy algorithm for offline Facility Location of [26], and cast its analysis in the framework of dual-fitting. Thus, they prove the slightly more general claim that $\text{cost}(u_i)$'s divided by $2H_n - 1$ comprise a feasible solution to (DP) (see [35, Section 4] for the details).

We are not aware of any examples showing that the analysis above is tight. Thus, it remains open whether one can show a competitive ratio of $o(\log n)$ for PD-OFL by a more careful analysis. It would be really interesting if one could establish a competitive ratio of $O(\frac{\log n}{\log \log n})$ for PD-OFL, thus showing that this simple deterministic algorithm achieves an asymptotically optimal competitive ratio.

3.3 An Asymptotically Optimal Deterministic Algorithm

We proceed to discuss a deterministic online algorithm with an asymptotically optimal competitive ratio. For simplicity, we focus on the special case of uniform facility costs.

The high level idea is to formulate a deterministic algorithm that has the main properties employed in the analysis of RandOFL, namely (i) that the current phase ends as soon as an inner demand opens a new facility, and (ii) that the total cost due to the inner demands by the moment when the first of them opens a new facility is at most $2f$. Then, we could somehow "simulate" the analysis of RandOFL and end up with a competitive ratio of $\Theta(\frac{\log n}{\log \log n})$. Unfortunately, it is not clear (whether and) how a deterministic algorithm could satisfy properties (i) and (ii). So, the deterministic algorithm of [19], or DetOFL in short, is formulated so as to satisfy appropriately relaxed versions of them.

DetOFL maintains its facility configuration $F_0 = \emptyset, F_1, \ldots, F_n$ and a set L of *unsatisfied demands* in response to the demand sequence $u_1, \ldots, u_n$. When demand u_i arrives, u_i is marked as unsatisfied and is added to L. DetOFL computes u_i's unsatisfied neighborhood $L(u_i)$, which consists of the unsatisfied demands located much closer to u_i than to any algorithm's facility, and the potential $\text{Pot}(L(u_i))$ accumulated in $L(u_i)$. Formally, $L(u_i) = \text{Ball}(u_i, d(F_{i-1}, u_i)/x) \cap L$, where x is a constant chosen sufficiently large, e.g., $x = 10$, and $\text{Pot}(L(u_i)) = \sum_{u \in L(u_i)} d(F_{i-1}, u)$. If the potential accumulated in u_i's unsatisfied neighborhood is at least f, DetOFL opens a new facility w at an appropriately selected location in $L(u_i)$. If $d(F_{i-1}, u_i) \geq f$, the new facility w is located at u_i. Otherwise, the new facility w is located at the center of the smallest radius ball of $L(u_i)$ that contains more than half of $L(u_i)$'s potential[3]. If a new facility w opens, the demands in $L(u_i)$ are marked as *satisfied* and are removed from L. If $\text{Pot}(L(u_i)) < f$, no new facilities open. Finally, u_i is assigned to the nearest facility in F_i.

[3] To achieve a sub-logarithmic competitive ratio, a deterministic algorithm must carefully select the location of the new facility. In [19, Section 6.1], it is shown that any algorithm similar to DetOFL which opens a new facility at u_i (or at some arbitrary location in $L(u_i)$) must have a competitive ratio of $\Omega(\log n)$. We also highlight that DetOFL's choice of the center of the smallest radius ball that contains most of $L(u_i)$'s potential is conceptually similar (though technically different) to PD-OFL's choice of the location with the maximum potential surplus.

Main Properties and Intuition. DetOFL employs the notion of unsatisfied demands to ensure that each demand u_i can increase the algorithm's facility cost at most once and by at most $d(F_{i-1}, u_i)$. Each demand u_i becomes unsatisfied upon arrival. As long as it remains unsatisfied, u_i holds a potential equal to its distance to the current algorithm's configuration, which is at most $d(F_{i-1}, u_i)$. When it is included in an unsatisfied neighborhood that opens a new facility, u_i is marked as satisfied, loses its potential, and cannot increase the algorithm's facility cost anymore.

To sketch the analysis of DetOFL, we use the notion of a small ball (or small neighborhood) to refer to a set of locations which are much closer to each other than to the nearest algorithm's facility. The diameter of a small ball is at most $1/\lambda$ times the distance of its center to the nearest facility, where λ is a constant chosen sufficiently larger than x, e.g., $\lambda = 2x + 1$. Intuitively, the demands of a small ball can be regarded as essentially located at the center of the ball.

Each demand u in a small ball B includes all unsatisfied demands in B in its unsatisfied neighborhood $L(u)$. Hence, the potential accumulated in any small ball B does not exceed the facility opening cost f. Otherwise, a new facility would have opened in B when its last demand arrived, and B's potential would have dropped to 0. This is the main invariant maintained by DetOFL, and can be regarded as a relaxed version of $\mathrm{RandOFL}$'s property (ii) above.

In fact, the invariant of DetOFL is also a relaxed version of the invariant maintained by PD-OFL. The idea is that DetOFL maintains its facility configuration so that after each demand u_i is processed, the distances $d(F_i, u_\ell)$, $\ell \leq i$, respect the dual constraints, but only at a local level. Since the analysis mostly cares about the algorithm's cost due to inner demands, which occupy a small ball around an optimal center, the facility opening rule of DetOFL ensures that each dual constraint is not violated if only the "local" demands, namely the demands in a small ball around each potential facility location, are taken into account in the constraints of (DP). It turns out that maintaining dual feasibility only locally facilitates the proof of an asymptotically optimal competitive ratio for DetOFL.

The second basic property of DetOFL has to do with the location of a new facility. A new facility w opened by a demand u_i is located in the smallest radius ball of $L(u_i)$ that contains more than half of $L(u_i)$'s potential. Therefore, if more than half of $L(u_i)$'s potential is included in a small ball, w is located very close to it. In particular, if some unsatisfied inner demands included in $L(u_i)$ contribute more than half of $L(u_i)$'s potential, w is located so close to them that the current phase ends. This property can be regarded as a relaxed version of $\mathrm{RandOFL}$'s property (i) above.

Competitive Analysis. We proceed to show how DetOFL's relaxed versions of $\mathrm{RandOFL}$'s properties (i) and (ii) imply an asymptotically optimal competitive ratio. Similarly to the analysis of $\mathrm{RandOFL}$, we consider a pair of positive integers h, m with $m^h > n$, and focus on a single optimal center c and the demands assigned to it in the optimal solution. We break down the analysis into $h + 2$ *disjoint phases* according to the distance of the algorithm's facility configuration to c. The j-th phase, $j = h, h-1, \ldots, 0$, begins just after a facility within a distance of $\lambda m^{j+1}\delta^*$ to c opens, and ends as soon as a facility within a distance of $\lambda m^j \delta^*$ to c opens. There is also a final phase, following phase 0, that never ends. The demands which either arrive in the current phase or remain unsatisfied from the previous phases are classified into *inner* and *outer* demands. A demand u is inner for phase j if $d^*_u < m^j\delta^*$ and outer otherwise (see also Fig. 4). For the first phase h, all demands are considered as inner, and for the final phase, all demands are considered as outer.

The distance of an outer demand u to c is so large that both its potential and its assignment cost can be bounded in terms of d^*_u. Thus, we can show that the total potential allocated to the outer demands and the total algorithm's assignment cost due to them are both at most $(\lambda(m+1)+1)\mathrm{Asg}^*(c)$.

By definition, the inner demands for any phase j are included in a small ball around c. For simplicity, we refer to the potential of unsatisfied inner demands as the potential of c. The main invariant

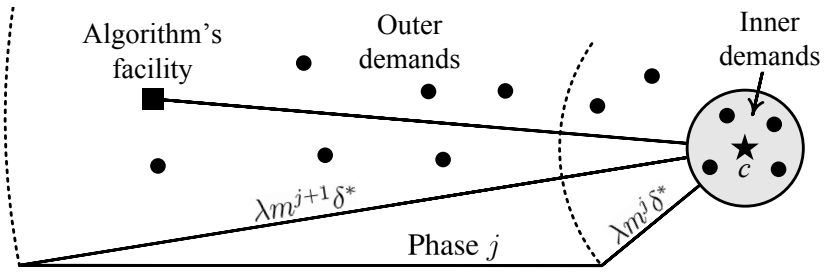

Fig. 4. In the analysis of DetOFL, phase j starts just after a facility in $\mathrm{Ball}(c, \lambda m^{j+1}\delta^*)$ opens, and ends as soon as a facility in $\mathrm{Ball}(c, \lambda m^j\delta^*)$ opens. The demands in $\mathrm{Ball}(c, m^j\delta^*)$, namely in the small grey ball around c, are *inner* and the remaining demands are *outer*. Choosing λ sufficiently large, we ensure that (i) each new inner demand includes all unsatisfied inner demands in its unsatisfied neighborhood, and that (ii) if more than half of the potential for a new facility w is contributed by unsatisfied inner demands, w is located in $\mathrm{Ball}(c, \lambda m^j\delta^*)$ and phase j ends.

maintained by DetOFL implies that the potential of c never exceeds f. Based on this property, we can use a potential function argument and bound the algorithm's facility cost and the algorithm's assignment cost due to the inner demands. The idea is to trace the changes in the potential of c in each phase j. Let the potential of c be 0 at the beginning of any phase. Every time an inner demand u arrives and remains unsatisfied at its assignment time (i.e., u does not open a new facility), the potential of c increases by u's assignment cost. Since the potential of c drops to 0 at the end of phase j, the total assignment cost of the inner demands which arrive in phase j and remain unsatisfied at their assignment time is bounded from above by the total decrease in c's potential during phase j (see also Fig. 5).

The potential of c decreases only if a demand u_i opens a new facility w, and either w is closer to c than any facility in F_{i-1}, or some unsatisfied inner demands are included in $L(u_i)$ and become satisfied as soon as w opens (or both). Since the potential of c is non-negative and at most f, we charge the algorithm with a cost of f for any decrease in the potential of c due to a new facility w. Moreover, the algorithm incurs a facility cost of f for the new facility and a cost no greater than f/x for the assignment of u_i to w. Overall, there is a cost of at most $(2 + \frac{1}{x})f$ associated each new facility w.

If the inner demands in $L(u_i)$ contribute more than half of its potential, the new facility w is within a distance of $\lambda m^j\delta^*$ to c, and phase j ends. Otherwise, the potential of $L(u_i)$'s outer demands is at least $f/2$. Thus, the cost associated with w is charged to the decrease in the potential of the outer demands in $L(u_i)$, which is in turn charged to their optimal assignment cost. This is possible because after w opens, the demands in $L(u_i)$ become satisfied and are not charged with any additional cost.

Therefore, the total algorithm's cost due to the demands assigned to c is at most $(h+1)(2+\frac{1}{x})f + 2(\lambda(m+1)+1)\mathrm{Asg}^*(c)$. Setting $m = h = \frac{\log n}{\log\log n}$, we obtain a competitive ratio of $\Theta(\frac{\log n}{\log\log n})$ against solutions consisting of a single center.

Multiple Optimal Centers. Unfortunately, DetOFL does not allow for an easy generalization of the analysis above to the case of $k > 1$ optimal centers $c_1, \ldots, c_k$. As before, the potential of each optimal center never exceeds f. However, a single new facility w may now (significantly) decrease the potential of many optimal centers at the same time. Hence, if we directly apply the argument above, we may end up with a cost of up to $\Omega(kf)$ associated with a new facility w. Such a large cost cannot be charged to the potential of the unsatisfied demands opening w, as it was sketched above. Therefore, bounding the decrease in the potential of each optimal center separately can only lead to a logarithmic upper bound on the competitive ratio.

To establish an asymptotically optimal competitive ratio, [19] develops an approach based on an appropriate hierarchical decomposition of the metric space consisting of $c_1, \ldots, c_k$. The decomposi-

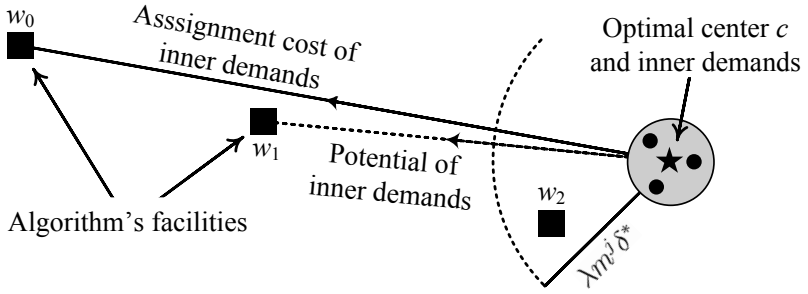

Fig. 5. Bounding the assignment cost of inner demands. Some inner demands arrive as long as w_0 is the nearest facility to the optimal center c and remain unsatisfied at their assignment time. Initially, the potential of each inner demand is equal to its assignment cost, which is roughly $d(w_0, c)$. As soon as a facility w_1 closer to c opens, the potential of each inner demand decreases to $d(w_1, c)$. The algorithm is charged with the decrease $d(w_0, c) - d(w_1, c)$ in the potential of each unsatisfied inner demand. When facility w_2 opens, the current phase ends. The algorithm is charged with the remaining potential $d(w_1, c)$ of each inner demand.

tion defines a hierarchy of optimal center *coalitions*. A coalition K is a set of centers that are much closer to each other than to any algorithm's facility. Therefore, as long as K forms a coalition, the total potential accumulated by (the unsatisfied inner demands associated with) the optimal centers in K is at most f. Hence the total decrease in the potential of the centers in K due to a new facility is at most f. Intuitively, we can think of (and the analysis can treat) K as a single optimal center.

Then, the analysis distinguishes between *isolated* and *non-isolated* coalitions. A coalition K is isolated if K is significantly closer to some algorithm's facility than to any optimal center outside K. Therefore, given a collection of disjoint isolated coalitions, a new facility can be closer to, and thus can decrease the potential of, at most one isolated coalition. Thus, a collection of disjoint isolated coalitions can be analyzed independently of each other and similarly to the special case where there is a single optimal center. This implies that DetOFL remains $\Theta(\frac{\log n}{\log \log n})$-competitive against solutions consisting of many optimal centers, as long as they can be partitioned into disjoint isolated coalitions. The key property of the hierarchical decomposition employed in [19] is that no coalition stays non-isolated for too long. Using this property, one can show that non-isolated coalitions only increase the competitive ratio by an additive constant term.

Non-Uniform Facility Costs. The generalization of DetOFL to non-uniform facility costs is more involved. It employs the notion of facility types and rounds down the facility costs to the nearest integral power of 2. When a demand u_i arrives, DetOFL marks u_i as unsatisfied, determines u_i's unsatisfied neighborhood $L(u_i)$, and calculates the potential of $L(u_i)$. The idea of the facility opening rule is to open a facility of *smallest type* in a small neighborhood around u_i, provided that its opening cost is no greater than $L(u_i)$'s potential. If there are many facilities of the same (smallest) type sufficiently close to u_i, DetOFL opens the one closest to the center of the smallest radius ball of $L(u_i)$ which contains more than half of $L(u_i)$'s potential. If u_i is the only demand in its unsatisfied neighborhood, a sequence of facilities approaching u_i may open. Generalizing the analysis for uniform facility costs, one can show that DetOFL is $\Theta(\frac{\log n}{\log \log n})$-competitive for non-uniform facility costs [19, Theorem 3].

3.4 A Simple Deterministic Algorithm for the Plane

DetOFL's careful selection of each new facility's location allows for an asymptotically optimal competitive ratio, on the one hand, but affects the algorithm's efficiency on the other. In fact, DetOFL

may need to maintain the locations of up to $\Omega(n)$ unsatisfied demands and may need up to $\Omega(n^2)$ time to process the n-th demand (see also the discussion in [3, Section 2]). Therefore, despite being a significant contribution towards understanding Online Facility Location from a theoretical point of view, DetOFL is of limited practical applicability.

Motivated by time and space efficiency considerations, Anagnostopoulos, Bent, Upfal, and Van Hentenryck [3] formulated a simple, natural, and computationally efficient deterministic online algorithm for Facility Location on the plane (and on spaces of bounded dimension, in general). The algorithm of Anagnostopoulos et al., or $\mathrm{PartOFL}$ in short, achieves a competitive ratio of $\Theta(\log n)$ for the plane and of $\mathrm{O}(2^d \log n)$ for d-dimensional spaces.

The operation of $\mathrm{PartOFL}$ follows the hierarchical structure of a quadtree. In particular, $\mathrm{PartOFL}$ assumes a hierarchical decomposition of the plane into quadrants with $\log_2 n$ levels. Level 0 consists of non-overlapping squares of side length f, which we call level-0 quadrants. Each level-j quadrant has a side length of $2^{-j}f$ and is partitioned into four level-$(j+1)$ children quadrants of side length $2^{-(j+1)}f$. The quadrants are classified into *open*, which have a facility at their center, *active*, which have an open parent and wait for more demands to be associated with them in order to become open, and *inactive*, which have yet to become active. Formally, a level-0 quadrant is active if there are no demands associated with it, and open otherwise. For any $j \geq 1$, if the parent of a level-j quadrant is open, the quadrant is active if there are less than 2^{j+2} demands associated with it, and open otherwise. If the parent of a level-j quadrant is either active or inactive, the quadrant is inactive.

$\mathrm{PartOFL}$ maintains a partitioning of the plane into active quadrants and associates each new demand with the active quadrant including it. When a new demand u arrives, $\mathrm{PartOFL}$ finds the active quadrant q_u in which u is included, and associates u with q_u. Let j_u be the level of q_u. If either $j_u = 0$ or 2^{j_u+2} demands have been associated with q_u, a new facility opens at the center of q_u and u is assigned to it. Furthermore, q_u becomes open and its children become active. Otherwise, q_u remains active and u is assigned to the facility at the center of q_u's parent.

Intuition. The high level idea of $\mathrm{PartOFL}$ is the same as the idea behind $\mathrm{RandOFL}$ and DetOFL. Namely, whenever the (facility opening) potential in a given area becomes large enough, the algorithm opens a new facility there, and starts tracking the potential in its subareas. Thus, at the conceptual level, all the three algorithms converge to the optimal centers by (implicitly or explicitly) considering a hierarchical decomposition of the metric space, and opening new facilities in smaller and smaller areas, provided that they include a sufficiently large number of demands (once more, it is worth observing the resemblance to the lower bound construction in Section 2). However, $\mathrm{RandOFL}$ and DetOFL are meant to work for general metric spaces, so they cannot explicitly rely on any fixed hierarchical decomposition. Thus DetOFL has to calculate explicitly the potential in the neighborhood of each new demand, which turns out to be computationally expensive, and $\mathrm{RandOFL}$ resorts to randomization and implements the same idea, though in a simpler and more efficient way. On the other hand, $\mathrm{PartOFL}$ employs a quadtree, fixed a-priori, and only tracks the potential accumulated in active quadrants. Relying on a fixed (and particularly simple) hierarchical decomposition makes the algorithm simpler to implement, considerably faster, and much more space efficient than DetOFL. In fact, $\mathrm{PartOFL}$ enjoys the conceptual simplicity and the time and space efficiency of $\mathrm{RandOFL}$ without resorting to randomization.

Competitive Analysis. $\mathrm{PartOFL}$ maintains a partitioning of the plane into active quadrants. Each active quadrant at level $j \geq 1$ has an open facility at one of its corners. Therefore, if a demand u is associated with an active quadrant at level j, u's assignment cost is at most $2^{-j}\sqrt{2}f$. Since each level-j active (resp. open) quadrant has less than (resp. exactly) 2^{j+2} demands associated with it, the assignment cost due to them is at most $4\sqrt{2}f$.

A competitive ratio of $O(\log n)$ follows by observing that there is an optimal center close to each open quadrant. More specifically, if q is an open quadrant at level j, we show that there is an optimal center either in q or in one of the eight level-j quadrants around q. For sake of contradiction, let us assume that there is no optimal center in any of them. Then the optimal assignment cost due to the demands associated with q is greater than f if $j = 0$, and greater than $2^{j+2} \cdot 2^{-j} f = 4f$ otherwise. If $j = 0$, opening a facility at the location of the demand associated with q costs f and decreases the demand's assignment cost to 0. If $j \geq 1$, opening a facility at the center of q costs f and decreases the assignment cost for the demands associated with q to $2\sqrt{2}f$. In both cases, opening a facility in q improves the cost of the optimal solution, which is a contradiction.

Therefore, there are at most 9 open quadrants per optimal center at each level. For each of them, the algorithm incurs a facility cost of f and an assignment cost of at most $4\sqrt{2}f$ due to the demands associated with it. In addition, each open quadrant has at most 4 active children quadrants at the next level. For each of them, the algorithm incurs an assignment cost of at most $4\sqrt{2}f$ due to the demands associated with it. Therefore, PartOFL incurs a total cost of $O(f)$ per level and per optimal center. Since we can have open and active quadrants in at most $\log_2 n$ levels, PartOFL's total cost is at most $O(\log n)$ times the optimal cost.

Additional Properties. Anagnostopoulos et al. show how PartOFL generalizes to the case where facilities can open only at the demands' locations, and to the case where facilities can open only at a fixed set of locations given to the algorithm in advance. The latter is a special case of non-uniform facility costs, where the opening cost for each location is either f or ∞. In both cases, PartOFL's competitive ratio for the plane is $O(\log n)$. Moreover, Anagnostopoulos et al. prove that PartOFL is $O(1)$-competitive, with high probability, if the demands are nearly uniformly distributed over the plane, and that for any instance, PartOFL has more or less the same competitive ratio against all demand orderings. Hence, PartOFL lacks a remarkable property of RandOFL, namely that for any instance, RandOFL is $O(1)$-competitive if the demands arrive in random order.

On the practical side, [3] presents an interesting experimental analysis of RandOFL, DetOFL, and PartOFL. The main message is that after some fine tuning of its parameters, DetOFL slightly outperforms PartOFL and RandOFL as far as the solution quality is concerned, but it is much more demanding computationally and much more difficult to implement. On the other hand, PartOFL achieves a better balance between computational complexity and solution quality, and consistently outperforms RandOFL.

4 An Incremental Algorithm for Facility Location

We proceed to discuss the incremental variant of Facility Location. In addition to opening new facilities and assigning new demands to them, an incremental algorithm can merge existing facilities (and the corresponding demand clusters) with each other. Hence, the irrevocable decisions made by an incremental algorithm concern only the clusters of demands assigned to the same facility, rather than the exact location of each facility. Rather surprisingly, [16] proved that for uniform facility costs, the incremental version of Facility Location admits a constant competitive ratio. In this section, we present the incremental algorithm of [16] and sketch its competitive analysis.

The algorithm of [16], or IncrFL in short, maintains its facility configuration $F_0 = \emptyset, F_1, \ldots, F_n$ and an assignment of the demands processed so far to its facilities in response to the demand sequence $u_1, \ldots, u_n$. For each facility $z \in F_i$, IncrFL maintains z's merge radius $m(z)$, the set $C(z)$ of the demands currently assigned to z, and the set $\text{Init}(z) \subseteq C(z)$ of the demands assigned to z upon arrival. Just after each demand u_i is processed, the union of $C(z)$ over all $z \in F_i$ must be $\{u_1, \ldots, u_i\}$.

Facility Opening Rule. IncrFL employs a simplified version of the facility opening rule of DetOFL. When a demand u_i arrives, it becomes unsatisfied and is added to the set of unsatisfied demands L. IncrFL computes u_i's unsatisfied neighborhood $L(u_i) = \mathrm{Ball}(u_i, d(F_{i-1}, u_i)/x) \cap L$, where x is a sufficiently large constant, and its potential $\mathrm{Pot}(L(u_i)) = \sum_{u \in L(u_i)} d(F_{i-1}, u)$.

If $\mathrm{Pot}(L(u_i)) \geq \beta f$, where β is a sufficiently large constant, IncrFL opens a new facility w at the location of u_i. Then w's merge radius $m(w)$ is initialized to $3d(F_{i-1}, w)/x$, and $C(w)$ and $\mathrm{Init}(w)$ are initialized to $\emptyset$. Moreover, the demands in $L(u_i)$ are marked as satisfied and are removed from L. If $\mathrm{Pot}(L(u_i)) < \beta f$, no new facilities open. Finally, u_i is assigned to the nearest facility $w_i \in F_i$ (if u_i opens a new facility w, w_i is simply w), and is added to $C(w_i)$ and to $\mathrm{Init}(w_i)$.

Facility Merge Rule. IncrFL employs an elegant merge rule which ensures that each facility $z \in F_i$ is the only facility in its merge ball $\mathrm{Ball}(z, m(z))$. If demand u_i opens a new facility w, every facility $z \in F_{i-1}$ that includes w in its merge ball $\mathrm{Ball}(z, m(z))$ is merged with w. Namely, every facility $z \in F_{i-1}$ with $d(z, w) \leq m(z)$ is closed and the demands assigned to it are reassigned to w (i.e., for each such z, $F_i = F_i \setminus \{z\}$ and $C(w) = C(w) \cup C(z)$).

IncrFL carefully adjusts the merge radius of its facilities. When a new facility w is opens, $m(w)$ is initialized to $3d(F_{i-1}, w)/x$, namely to thrice the radius of the unsatisfied neighborhood opening w. Furthermore, as more and more demands are initially assigned to w, the algorithm may decrease $m(w)$ so as to ensure that no merge operation can dramatically increase the assignment cost of the demands in $\mathrm{Init}(w)$. In particular, IncrFL maintains the invariant that the merge radius $m(z)$ of any facility z satisfies that:

$$|\mathrm{Init}(z) \cap \mathrm{Ball}(z, m(z)/4)| \cdot m(z) \leq \beta f \tag{2}$$

So after demand u_i is assigned to w_i, the algorithm checks whether $m(w_i)$ satisfies (2). If not, $m(w_i)$ decreases to the largest value that satisfies (2).

Main Properties and Intuition. To discuss the main properties and sketch the analysis[4] of IncrFL, we use the notation introduced for DetOFL, in Section 3.3. Similarly to DetOFL, IncrFL uses the notion of unsatisfied demands and ensures that each demand u_i can increase the algorithm's facility cost at most once and by at most $d(F_{i-1}, u_i)$. Moreover, the facility opening rule ensures that the potential accumulated in any small ball is at most βf. Choosing β sufficiently large, we ensure that each new facility w is significantly closer to some optimal center than any of the existing facilities. More precisely, let u_i be a demand opening a new facility w, and let c be the nearest optimal center to w. One can show that if $\beta \geq 7$, then $d(c, w) \leq d(F_{i-1}, c)/3$.

An important property of the merge rule is that for every facility w, there always exists a facility in $\mathrm{Ball}(w, \frac{x}{x-3}m(w))$ and the demands currently assigned to w remain assigned to facility in $\mathrm{Ball}(w, \frac{x}{x-3}m(w))$. This holds because when an existing facility w is merged with a new facility w', the merge radius of w' is at most $3/x$ times the merge radius of w. Therefore, the merge radii along any sequence of merge operations are geometrically decreasing (see also Fig. 6.a).

At the conceptual level, instead of pinning each new facility w down to a single location, IncrFL (irrevocably) determines an entire area, namely $\mathrm{Ball}(w, \frac{x}{x-3}m(w))$, in which a facility should exist (e.g., because the number of demands in this area is so large that an optimal center must be included in it). At the beginning, the algorithm does not have enough information about the demand sequence, and thus the area covered by each facility may (and should) be quite large. This helps the algorithm

[4] Focusing on the Euclidean plane, one may obtain a rather accurate picture of IncrFL by thinking of it as an incremental version of PartOFL that has facilities only in undominated open quadrants, namely in open quadrants that do not include any smaller open quadrants. When an active child quadrant of an undominated open quadrant becomes open, the facility in the parent quadrant is merged with the new facility in the child quadrant, and the parent quadrant becomes dominated by its open child. Of course, there are many substantial technical details hidden behind this intuitive description.

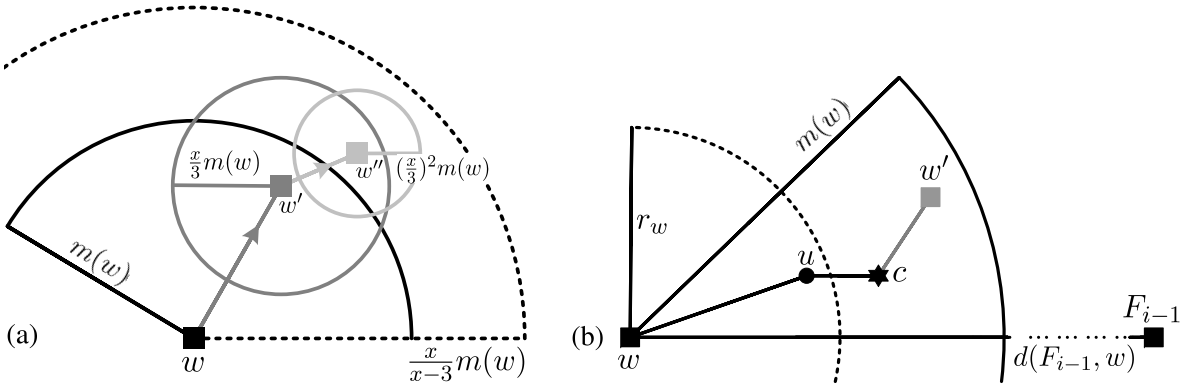

Fig. 6. (a) If w is merged with a new facility w', w is open when w' opens, and thus the merge radius $m(w')$ of w' is at most $3d(w, w')/x \leq 3m(w)/x$. If w' is subsequently merged with a new facility w'', $m(w'') \leq \frac{3}{x}m(w') \leq (\frac{3}{x})^2 m(w)$, and so on. Therefore, the demands currently assigned to w remain assigned to facility in $\mathrm{Ball}(w, \frac{x}{x-3}m(w))$. (b) Let w be an unsupported facility opened by demand u_i. For simplicity, we let $r_w \equiv d(F_{i-1}, w)/x$ denote the radius of $L(u_i)$. By the definition of unsupported facilities, there is a demand $u \in L(u_i)$ and an optimal center c at distance $d(c, u) \leq d(F_{i-1}, u)/(3x)$. Since $d(w, u) \leq r_w$, i.e., u is located much closer to w than to any other facility, $d(F_{i-1}, u) \approx d(F_{i-1}, w)$. Therefore, $d(c, u)$ is bounded from above by roughly $r_w/3$, and $d(c, w)$ is bounded from above by roughly $4r_w/3$. Thus, $m(w) = 3r_w > 2d(c, w)$. Any new facility w' for which c is the nearest optimal center has $d(c, w') \leq d(c, w)/3 < m(w)/6$. Consequently, $d(w, w') \leq m(w)$, and w is merged by the moment when w' opens.

to keep its facility cost reasonably close to the optimal cost, but as more and more demands arrive, it may lead to a dramatic increase in the algorithm's assignment cost due to some merge operation. Thus, as more and more demands arrive, the algorithm restricts facility w to smaller and smaller areas either by decreasing $m(w)$ according to (2) or by merging w with some new facility w', which has a geometrically smaller merge radius and fulfills $\mathrm{Ball}(w', \frac{x}{x-3}m(w')) \subseteq \mathrm{Ball}(w, \frac{x}{x-3}m(w))$.

The intuition above naturally leads to a new notion of distance, the so-called *extended distance*, that is the distance of a location p to the most distant location in the area covered by some facility. Formally, the extended distance of a location p to the algorithm's configuration F_i is:

$$g(F_i, p) = \min_{z \in F_i} \Big\{ d(z, p) + \tfrac{x}{x-3} m(z) \Big\}$$

For the analysis, it is particularly useful that the extended distance of p is non-increasing with time (note that the distance of a location p to the nearest facility may increase due to some merge operation). In particular, we use the *extended assignment cost* of a demand u, namely u's extended distance to the current algorithm's configuration, and bound u's assignment cost at any future point in time.

Competitive Analysis. In the analysis of online algorithms, we observe a recurrent pattern: In each phase, the algorithm incurs an assignment cost of roughly f due to the inner demands arriving before there is a facility close to them, and a facility cost of f for the first such facility. On the other hand, the algorithm's cost due to the outer demands can be easily bounded in terms of their optimal assignment cost. So, the reason that online algorithms end up with an (almost) logarithmic competitive ratio is that they incur a cost of $\mathrm{O}(f)$ due to the inner demands in each phase (this becomes evident if one applies the analysis of $\mathrm{RandOFL}$ and DetOFL for $m = 2$ and $h = \log n$). Moreover, the lower bound in Section 2 demonstrates that this is somehow unavoidable for any online algorithm.

Facility Cost. The observation above gives an indication about how the analysis should proceed, at least as far as the facility cost is concerned. We need not worry about facilities opened by unsatisfied

demands whose potential is large enough compared against their optimal assignment cost. Hence, if a demand u_i opens a new facility w and every demand u in $L(u_i)$ has $d_u^* \geq d(F_{i-1}, u)/(3x)$, we say that w is a *supported* (by the optimal solution) facility. Then, $3x \sum_{u \in L(u_i)} d_u^* \geq \mathrm{Pot}(L(u_i)) \geq \beta f$, and the opening cost of w can be charged to the optimal assignment cost of the unsatisfied demands in $L(u_i)$. Since each demand contributes to the facility cost at most once, the total cost of supported facilities is at most $3x\mathrm{Asg}^*/\beta$.

On the other hand, we should be concerned about *unsupported* facilities, namely facilities not fulfilling the criterion above. Intuitively, unsupported facilities correspond to the facilities opened by unsatisfied inner demands, and there may be up to a logarithmic number of them per optimal center. However, the merge rule of IncrFL succeeds in eliminating all but the most recent of them. By definition, the unsatisfied neighborhood opening an unsupported facility w includes a demand u with $d_u^* < d(F_{i-1}, u)/(3x)$. Namely, u is located $3x$ times closer to an optimal center c than to any algorithm's facility. Using that u lies in w's neighborhood, one can prove that w's distance to c is bounded from above by roughly $\frac{4}{3}d(F_{i-1}, w)/x$, and thus w's merge radius $m(w)$, which is initialized to $3d(F_{i-1}, w)/x$, is more than twice as large as w's distance to c (and this cannot change because of (2)). In simple words, the merge ball of any unsupported facility w includes a sufficiently large area around an optimal center c. Since each new facility w' for which c is the nearest optimal center is at least 3 times closer to c than w, w' is included in w's merge ball, and thus w is merged by the moment when w' opens (see also Fig. 6.b). Therefore, the merge rule ensures that at any point in time, there is at most one unsupported facility per optimal center. Putting the cost of supported and unsupported facilities together, we obtain that IncrFL's facility cost never exceeds $\mathrm{Fac}^* + 3x\mathrm{Asg}^*/\beta$.

Assignment Cost. Bounding IncrFL's assignment cost is technically involved. The actual reason comes from the lower bound in Section 2. A careful look at the lower bound construction reveals that any algorithm that processes the demands as they arrive and maintains $\mathrm{o}(\log n)$ facilities per optimal center must incur a total cost for the demands' initial assignment of $\omega(1)$ times the optimal cost, where the initial assignment cost of a demand u_i is $d(u_i, w_i)$, namely equal to u_i's distance to the first facility to which u_i is assigned. Therefore, to establish a constant competitive ratio, the analysis has to establish that in addition to the facility cost, the merge rule succeeds in decreasing the assignment cost to a constant times the optimal cost.

As in Sections 3.1 and 3.3, the analysis of the assignment cost distinguishes between inner and outer demands. But now the analysis is not divided into phases, and inner and outer demands are defined in an (intuitively similar but) slightly more general way. More precisely, a demand is outer if its initial assignment cost is within a constant factor of its optimal assignment cost, and inner otherwise. Using the notion of extended assignment cost, one can show that despite the merge operations, the assignment cost of an outer demand remains within a constant factor of its optimal assignment cost.

The main concern of the analysis is to bound the total assignment cost of inner demands throughout the execution of the algorithm. The idea is to show that in a sequence of merge operations, the assignment cost of (most of) the inner demands involved keeps converging to their optimal assignment cost. For simplicity, we first sketch the main idea of the analysis for a simple optimal center c and the inner demands assigned to it in the optimal solution.

The analysis further distinguishes between *good* and *bad* inner demands. At the intuitive level, an inner demand u starts as a good one, and remains good as long as its assignment cost converges to its optimal assignment cost. While an inner demand is good, we charge the algorithm with its actual assignment cost. More formally, let w be the facility which is currently the nearest one to c (if c is the only optimal center, w is just the most recent facility to open). A new inner demand is initially assigned to w because inner demands are essentially located at c, and because w is (by far) the nearest

facility to c. The facility opening rule ensures that the total initial assignment cost of the inner demands arriving as long as w is the nearest facility to c is at most βf.

At some point, a new facility w' opens. The facility opening rule ensures that w' is much closer to c than w, namely that $d(w', c) \leq d(w, c)/3$, and thus w' becomes the (new) nearest facility to c. If w is merged with w', the assignment cost of the inner demands assigned to w decreases at least by a factor of 2, because w' is much closer to c than w. Due to such a sequence of merge operations where the previous nearest facility to c is merged with the next nearest facility to c, some inner demands remain assigned to the nearest facility to c, and thus their assignment cost keeps converging to their optimal assignment cost. As long as this happens, the inner demands assigned to the nearest facility to c are classified as good inner demands. Using the argument above, one can show that the total assignment cost of good inner demands cannot exceed $2\beta f$ plus their optimal assignment cost.

On the other hand, if w, i.e., the previous nearest facility to c, is not merged with w', i.e., the next nearest facility to c, the inner demands assigned to c are classified as bad ones. When some inner demands become bad, they are irrevocably charged with their extended assignment cost, which is an upper bound on their actual assignment cost at any future point in time. Using (2) and the fact that w's merge radius is rather small, one can show that the total extended assignment cost of the inner demands which become bad when w is not merged with w' is $\mathrm{O}(\beta f)$ plus their optimal assignment cost. On the other hand, if w is not merged with w', w must be a supported facility. Therefore, the optimal assignment cost of the unsatisfied demands opening w is at least $\beta f/(3x)$, and can compensate for the extended assignment cost of the inner demands which become bad due to w. Moreover, since w' is much closer to c than w, no additional inner demands are assigned to w after w' opens.

The arguments above imply that for the special case of a single optimal center c, the assignment cost of IncrFL never exceeds a constant factor times the optimal cost. To generalize the analysis to multiple optimal centers, we employ the hierarchical decomposition approach developed in [19] for the analysis of DetOFL. Again, we consider coalitions of optimal centers and distinguish between isolated and non-isolated coalitions. For IncrFL, (isolated and non-isolated) coalitions are defined with respect to their extended distance to the algorithm's configuration, where it is crucial that the extended distance is non-increasing with time.

The idea is to show that any collection of disjoint isolated coalitions can be analyzed independently of each other and similarly to the special case where there is a single optimal center. This implies that IncrFL is $\mathrm{O}(1)$-competitive against solutions consisting of many optimal centers, as long as they can be partitioned into disjoint isolated coalitions. For non-isolated coalitions, one can employ essentially the analysis of DetOFL, and show that since no coalition stays non-isolated for too long, non-isolated coalitions only increase the competitive ratio of IncrFL by an additive constant term.

5 A Relaxed Incremental Model for Facility Location

A relaxed variant of Incremental Facility Location was studied in [17, 13]. In this variant, the irrevocable decisions made by the algorithm concern any increase in the algorithm's facility cost rather than the assignment of some demands to the same facility. Hence, in addition to opening new facilities and merging existing facilities with each other, the algorithms of [17, 13] can also reassign demands to the nearest facility. However, the algorithm is charged for its maximum facility cost over time, and thus increasing its facility cost is an irrevocable decision.

5.1 A Memoryless Algorithm with Constant Competitive Ratio

The algorithm of [17] is essentially a streaming algorithm for Facility Location. Similarly to online and incremental algorithms, *streaming algorithms* have only sequential access to the input and typi-

cally make a single pass over the input sequence. However, instead of irrevocable decisions to which online and incremental algorithms commit themselves, the emphasis of streaming algorithms is on computational efficiency. The goal of a streaming algorithm is to compute a good approximate solution in space typically polylogarithmic in the input size, while spending a typically polylogarithmic processing time per input element.

For Facility Location however, one cannot hope for a streaming algorithm that maintains an $\mathrm{O}(1)$-approximate facility configuration and is really efficient in the worst case. The reason is that there are instances where every (near-)optimal solution needs to open $\Omega(n)$ facilities, and every $\mathrm{O}(1)$-approximation algorithm needs to run for $\Omega(n^2)$ time (see e.g., [42, Section 4]). Thus, Indyk [25] suggested that an algorithm may maintain only the cost of an approximate facility configuration and not the configuration itself. For the discrete d-dimensional space, Indyk [25] presented a randomized algorithm that approximates the optimal cost within a factor of $\mathrm{O}(d \log^2 n)$ in $O(n\mathrm{poly}(\log n))$ time and $\mathrm{O}(d^2 \log^2 n)$ space. Subsequently, the approximation ratio was improved to a constant by Lammersen and Sohler [28]. It is worth mentioning that the cost estimator used in [25] is conceptually similar to $\mathrm{PartOFL}$, and that the cost estimator used for the algorithm of [28] is conceptually similar to the incremental version of $\mathrm{PartOFL}$ discussed in Footnote 4.

Motivated by streaming applications that require an approximate facility configuration and not just an estimation of the optimal cost, [17] presented a Facility Location algorithm that maintains an $\mathrm{O}(1)$-approximate facility configuration and uses a minimal amount of computational resources. The algorithm of [17], or MemlFL in short, processes the demand points one-by-one, as they arrive, and keeps in memory only the locations of its facilities and some additional information of constant size per facility. So MemlFL is *memoryless*, in the sense that it maintains an approximate facility configuration using essentially just the space required for that. In this sense, $\mathrm{RandOFL}$ and $\mathrm{PartOFL}$ are also memoryless, but since they commit themselves to irrevocable decisions about their facility locations, they suffer from facility proliferation and an (almost) logarithmic competitive ratio.

The Memoryless Algorithm. MemlFL can be regarded as a randomized memoryless version of IncrFL. MemlFL maintains its facility configuration $F_0 = \emptyset, F_1, \ldots, F_n$ and the replacement radius $m(z)$ for each facility z in response to the demand sequence $u_1, \ldots, u_n$.

MemlFL is based on the notion of extended assignment cost, and uses the facility opening rule of $\mathrm{RandOFL}$ and a memoryless version of IncrFL's merge rule. The extended assignment cost of a demand u to a facility z is now $g(z, u) = d(z, p) + 2m(z)$. When a demand u_i arrives, MemlFL calculates u_i's extended assignment cost $g(F_{i-1}, u_i) = \min_{z \in F_{i-1}}\{d(z, u_i)\}$ with respect to the current configuration F_{i-1}. Demand u_i opens a new facility w located at u_i with probability $g(F_{i-1}, u_i)/(\beta f)$, where β is a small constant. If a new facility w opens, its replacement radius $m(w)$ is set to $\min\{g(F_{i-1}, u_i), \beta f\}/6$, and any other facility z that includes w in its replacement ball $\mathrm{Ball}(z, m(z))$ is replaced[5] by w. Namely, each $z \in F_{i-1}$ with $d(z, w) \leq m(z)$ is removed from F_i. Intuitively, w replaces all facilities whose removal from F_i does not increase the extended distance of any location. The total cost of MemlFL just after demand u_i is processed is:

$$\max_{1 \leq \ell \leq i}\{|F_i|\, f\} + \sum_{\ell=1}^{i} d(F_i, u_\ell) \tag{3}$$

Competitive Analysis. Combining ideas from the analyses of IncrFL and $\mathrm{RandOFL}$, one can show that the competitive ratio of MemlFL is less than 14. Using the notions of supported and unsupported

[5] Even though the facility replacement rule of MemlFL is almost identical to the facility merge rule of IncrFL, we use the term "replacement" because we wish to emphasize that when z is replaced by w, each demand assigned to z is reassigned to the nearest facility available (and not necessarily to w, as it happens when z is merged with w in IncrFL).

facilities, as in the analysis of IncrFL, one can show that the expected facility cost of MemlFL is at most Fac^* plus a (small) constant times Asg^*.

Though similar, the analysis of MemlFL's assignment cost is simpler than that of IncrFL. This happens not only because MemlFL is simpler than IncrFL, but mostly because the demands are now reassigned to the nearest facility without any restrictions. As in the analysis of IncrFL, the extended distance of any location to the algorithm's configuration is non-increasing with time. We use the extended distance of an optimal center c and break down the analysis of the assignment cost for the demands assigned to it into a number of disjoint phases, numbered $0, 1, \ldots$ as the time goes. For every integer $j \geq 0$, the j-th phase begins when the extended distance of c becomes less than $f/2^{j-1}$ and ends when the extended distance of c becomes less than $f/2^j$. A demand u arriving in phase j is inner if $d_u^* < f/(2^j\lambda)$, where λ is a constant chosen sufficiently large, and outer otherwise. Thus, when an outer demand u arrives, its extended assignment cost is at most $(2\lambda+1)d_u^*$. Thus the actual assignment cost of an outer demand u never exceeds $(2\lambda+1)d_u^*$.

Provided that λ is large enough, the inner demands arriving in the current phase can be regarded as essentially located at the optimal center c. Using this intuition, one can prove (i) that the current phase ends as soon as an inner demand opens a new facility, and (ii) that every inner demand arriving in phase j opens a new facility with probability roughly $1/(2^j\beta)$. Therefore, the expected number of inner demands arriving in phase j is roughly $2^j\beta$ and the expected number of inner demands arriving up to the end of phase j is roughly $2^{j+1}\beta$. As long as c is in phase j, the extended assignment cost of every demand is at most $f/2^{j-1}$ plus its optimal assignment cost. Therefore, the total expected extended assignment cost of the inner demands arriving up to the end of phase j (with respect to any algorithm's configuration in phase j) is roughly $4\beta f$ plus their optimal assignment cost.

Hence, the expected assignment cost of MemlFL for the demands assigned to c is bounded from above by roughly $4\beta f$ plus $(2\lambda+1)\mathrm{Asg}^*(c)$. Putting everything together and optimizing the choice of β and λ, we obtain that the competitive ratio of MemlFL is less than 14.

Non-Uniform Facility Costs. In [17, Section 3], it is shown that MemlFL generalizes to non-uniform facility costs and achieves a competitive ratio of less than 49.

5.2 Incremental Facility Location with Facility Moves

The model of Incremental Facility Location with facility moves of Divéki and Imreh [13] is a further relaxation of the model of [17]. The algorithm of [13] is not memoryless and can move its facilities anywhere when a new demand arrives, but as in [17], its is charged for its maximum facility cost over time, and thus increasing its number of facilities is an irrevocable decision.

The algorithm of [13], or MoveFL, maintains its facility configuration $F_0, F_1, \ldots, F_n$ in response to the demand sequence $u_1, \ldots, u_n$. When a demand u_i arrives, MoveFL uses an offline algorithm Alg_1 and solves the Facility Location instance for demands $u_1, \ldots, u_i$. Let F_i^* be the facility configuration computed by Alg_1 for the demands up to u_i. If $|F_{i-1}| \leq |F_i^*|$, MoveFL adopts F_i^* as its current facility configuration F_i. Otherwise, MoveFL solves the $|F_{i-1}|$-Median instance for the demands up to u_i by an offline algorithm Alg_2, and adopts the solution of Alg_2 as its current facility configuration F_i. The total cost of MoveFL just after demand u_i is processed is again given by (3).

Divéki and Imreh proved that the competitive ratio of MoveFL is $c_1(1+c_2)$, where c_1 (resp. c_2) is the approximation ratio of algorithm Alg_1 (resp. Alg_2) for Facility Location (resp. k-Median). If Alg_1 and Alg_2 are exact algorithms, [13] shows that MoveFL is 2-competitive for general metric spaces and $3/2$-competitive on the line. On the negative side, [13] shows that the competitive ratio of any incremental algorithm for the relaxed model with facility moves is at least 1.15, even if the metric space is a line segment.

6 Application to Streaming and Incremental Algorithms for k-Median

A few streaming algorithms for k-Median draw ideas and techniques from previous work on online and incremental algorithms for Facility Location, especially from the memoryless algorithms of Meyerson [33] and Anagnostopoulos et al. [3]. For instance, the best known streaming algorithm for k-Median on general metric spaces [10] uses $\mathrm{RandOFL}$ as its main building block (see also [23]), and the coreset construction employed in the streaming algorithm of [21] for k-Median on the discrete d-dimensional space is conceptually similar to the approach of $\mathrm{PartOFL}$. In this section, we discuss a generic way of turning any online or incremental algorithm for Facility Location into an incremental or streaming algorithm for k-Median with similar performance characteristics. We also discuss how this general approach is employed in the streaming algorithm of [10] and in the incremental algorithm of [16], where the main building blocks are $\mathrm{RandOFL}$ and IncrFL, respectively.

Generic Algorithm. The idea is to exploit the intrinsic similarity between Facility Location with uniform facility costs and k-Median. In fact, any feasible solution for an instance of k-Median can be regarded as a feasible solution for an instance of Facility Location on the same set of demands. Moreover, if the facility cost f is chosen appropriately, the Facility Location algorithm computes a solution with not much more than k facilities and an assignment cost reasonably close to the optimal assignment cost for the k-Median instance. We note that a few approximation algorithms for k-Median are based on the same idea (see e.g., [9, 27]), and that the same approach has been applied to the incremental algorithms for k-Center [8] and Sum k-Radius [11].

To sketch the main idea, we let Alg be any online or incremental algorithm for Facility Location. We assume that for any (uniform) facility cost f and any demand sequence, the facility and the assignment cost of the solution maintained by Alg are both at most $\psi\mathrm{Fac}^* + \rho\mathrm{Asg}^*$, where $\psi, \rho > 1$ are determined by Alg's competitive analysis (if Alg is a randomized algorithm, we consider its expected facility and assignment cost). For example, we can use $\psi = 1 + \log n$ and $\rho = 4$ for $\mathrm{RandOFL}$, $\psi = \Theta(\log n)$ and $\rho = \mathrm{O}(1)$ for DetOFL, and $\psi = \mathrm{O}(1)$ and $\rho = \mathrm{O}(1)$ for IncrFL and MemlFL.

Let $u_1, \ldots, u_n$ be any demand sequence, and let F^*, $|F^*| = k$, be an optimal k-median configuration and OPT be its assignment cost for $u_1, \ldots, u_n$. We consider the application of Alg to the demand sequence $u_1, \ldots, u_n$ with a facility cost of $f = \Lambda/(k\psi)$, where $\Lambda > 0$ is meant to be an estimation of OPT, and bound the number of facilities and the assignment cost of Alg. For the Facility Location instance processed by Alg, F^* gives a solution with a facility cost of Λ/ψ and an assignment cost of OPT. Therefore, the facility and the assignment cost of the solution maintained by Alg are both at most $\Lambda + \rho\mathrm{OPT}$. Since the cost for each facility is $\Lambda/(k\psi)$, the number of facilities maintained by Alg is at most $k\psi(1 + \rho\mathrm{OPT}/\Lambda)$. Thus, if Λ is within a constant factor of OPT, e.g., if $\mathrm{OPT} \le \Lambda \le \mu\mathrm{OPT}$, for some constant $\mu > 1$, Alg maintains a $(\rho + \mu)$-approximation to the optimal solution for the k-Median instance using at most $k\psi(1 + \rho)$ medians. The idea is to use doubling, a standard tool in online algorithms and competitive analysis (see e.g., [12]), and maintain incrementally an appropriate estimation Λ of OPT.

Hence, using Alg as a building block, we obtain an incremental (or streaming) algorithm for k-Median that operates in phases. Each phase i employs an estimation Λ_i of the optimal cost and invokes Alg with a facility cost of $\Lambda_i/(k\psi)$. The previous phase $i - 1$ ends with at most $k\psi(1 + \rho)$ medians whose assignment cost for the demand sequence processed in phase $i - 1$ is no greater than $(1+\rho)\Lambda_{i-1}$. The medians produced by phase $i-1$ are weighted by the number of demands assigned to each of them. Then phase i invokes Alg with a facility cost of $\Lambda_i/(k\psi)$ first on the weighted medians produced by phase $i - 1$ and then on the demands yet to be processed, as they arrive. Phase i lasts as long as the number of medians maintained by Alg is at most $k\psi(1 + \rho)$ and the assignment cost for the demand sequence processed in phase i is no greater than $(1 + \rho)\Lambda_i$. The first time that a new

demand u is about to increase either the number of medians above $k\psi(1+\rho)$ or the assignment cost above $(1+\rho)\Lambda_i$, phase i ends (without processing u), and phase $i+1$ begins with $\Lambda_{i+1} = (2\rho+3)\Lambda_i$.

Competitive Analysis. The total assignment cost of the algorithm above for the demands processed up to the end of phase $i-1$ to the medians produced by phase $i-1$ is:

$$\mathrm{cost}_{i-1} \leq (\rho+1)\Lambda_1 + (\rho+1)\Lambda_2 + \ldots + (\rho+1)\Lambda_{i-1}$$

The first term on the right-hand side is an upper bound on the cost for the assignment of the demands processed in the first phase to the weighted medians produced by the first phase, the second term is an upper bound on the cost for the assignment of the weighted medians of the first phase and the demands processed in the second phase to the weighted medians produced by the second phase, and so on. Since Λ's increase geometrically by a factor of $(2\rho+3)$, $\mathrm{cost}_{i-1} \leq \frac{2\rho+3}{2}\Lambda_{i-1} = \Lambda_i/2$.

Due to Alg's performance guarantee, when phase i ends, the optimal cost OPT'_i for the demand sequence processed in phase i (along with the last demand u) is greater than Λ_i. Of course, OPT'_i may be larger than OPT because the first part of the demand sequence processed in phase i consists of the weighted medians produced by phase $i-1$ (instead of the original demands assigned to them). But since the total distance of the original demands to the weighted medians produced by phase $i-1$ is at most cost_{i-1}, $\mathrm{OPT}'_i \leq \mathrm{OPT} + \mathrm{cost}_{i-1}$ (see e.g., [24, Theorem 2.3] for a detailed proof of this claim). Since $\Lambda_i < \mathrm{OPT}'_i$ and $\mathrm{cost}_{i-1} \leq \Lambda_i/2$, we obtain that $\Lambda_i < 2\mathrm{OPT}$ and that $\Lambda_{i+1} < 2(2\rho+3)\mathrm{OPT}$. Therefore, the assignment cost of the demands processed up to the end of phase $i+1$ to the medians maintained by phase $i+1$ is at most $\frac{2\rho+3}{2}\Lambda_{i+1} \leq (2\rho+3)^2\mathrm{OPT}$. Hence, the algorithm above maintains incrementally a configutation of at most $k\psi(1+\rho)$ medians with an assignment cost of $(2\rho+3)^2$ times the optimal assignment cost for the corresponding k-Median instance.

Streaming Algorithm for k-Median on General Metrics. Employing the generic algorithm above with $\mathrm{RandOFL}$ as the main building block, Charikar, O'Callaghan, and Panigrahy [10] obtained the best known streaming algorithm for k-Median on general metric spaces. The algorithm of Charikar et al., or Stream in short, achieves a constant approximation ratio, with polynomially high probability, and runs in $\mathrm{O}(k\log^2 n)$ space and in $\mathrm{O}(nk\log^2 n)$ time. Stream can also be regarded as an incremental algorithm for k-Median that achieves a constant competitive ratio using $\mathrm{O}(k\log^2 n)$ medians.

Stream follows the generic algorithm above with a subtle difference dictated by the fact that $\mathrm{RandOFL}$'s performance guarantee holds only on expectation. Thus, each phase i of Stream employs $\Theta(\log n)$ independent parallel invocations of $\mathrm{RandOFL}$, with each invocation ending when either its number of facilities becomes greater than $4k\psi(1+\rho)$ or its assignment cost becomes greater than $4(1+\rho)\Lambda_i$. Phase i ends when all invocations of $\mathrm{RandOFL}$ end, and the weighted medians produced by the invocation finished last are passed to the next phase.

By $\mathrm{RandOFL}$'s performance guarantee and Markov's inequality, the probability that an invocation of $\mathrm{RandOFL}$ in phase i ends before OPT'_i exceeds Λ_i is at most $1/2$. Therefore, the probability that any fixed phase i ends before OPT'_i exceeds Λ_i is polynomially small. Moreover, one can show that there are at most n phases, and thus, with polynomially high probability, all phases i end after their OPT'_i's exceed their Λ_i's. Except for this technical issue, the analysis of Stream is as above. The only difference is that Λ's should scale a bit faster, which increases the competitive ratio by a small constant factor. Since for $\mathrm{RandOFL}$, we can have $\psi = 1+\log n$ and $\rho = 4$, Stream maintains $\mathrm{O}(k\log^2 n)$ medians and achieves a constant competitive ratio with polynomially high probability.

Incremental Algorithm for k-Median. Employing the generic algorithm above with IncrFL as the main building block, [16] obtained a deterministic incremental algorithm for k-Median which achieves a constant competitive ratio using $\mathrm{O}(k)$ medians. Thus, [16] resolved an open question by Charikar

and Panigrahy [11], who proved that any deterministic incremental algorithm for k-Median that maintains at most $(1+\varepsilon)k$ medians must have a competitive ratio of $\Omega(1/\varepsilon)$, and posed as an open problem the existence of an incremental algorithm which maintains $\mathrm{O}(k)$ medians and achieves a constant competitive ratio.

Although the algorithm of [16] is an improvement on Stream with respect to the number of medians, it is much slower and dramatically more space consuming. An interesting open question is whether one can use the generic algorithm above with MemlFL (or with some other memoryless $\mathrm{O}(1)$-competitive algorithm for Facility Location). This could lead to an improved incremental / streaming algorithm for k-Median on general metric spaces with $\mathrm{O}(k \log n)$, or even $\mathrm{O}(k)$, medians (and a similar space consumption) and a running time of $\mathrm{O}(nk \log n)$, or even of $\mathrm{O}(nk)$. The main technical issue with MemlFL is that it does not maintain (a good estimation of) the demands' assignment cost with respect to the current facility configuration (see also the discussion in [17, Section 1]). Hence, it is not clear how one can determine whether the assignment cost in the i-th phase of the generic algorithm has exceeded $(1+\rho)\Lambda_i$ or not.

7 Other Applications and Related Work

We conclude this survey with a brief discussion of some work on approximation and online algorithms for Facility Location that either draws ideas and techniques from or is related to the work on online and incremental algorithms presented before.

Computational Efficiency. In the offline setting, where the demands are known in advance, one can apply $\mathrm{RandOFL}$ to a random permutation of the demands and obtain an $\mathrm{O}(1)$-approximate facility configuration in time $\mathrm{O}(n^2)$, i.e., linear in the size of the input. Although the approximation ratio is considerably worse than the best known approximation ratio for Facility Location, one can use $\mathrm{RandOFL}$ as a preprocessing step and improve the computational efficiency of some known algorithms. For instance, Meyerson [33, Section 5] showed how one can apply $\mathrm{RandOFL}$ as a preprocessing step and improve the running time of the local search algorithm of Charikar and Guha [9] from $\mathrm{O}(n^2/\varepsilon + n^2 \log n)$ to $\mathrm{O}(n^2/\varepsilon)$. Along the same lines, Guha et al. [23, Section 4.3] showed how a good initial solution computed by $\mathrm{RandOFL}$ on specific parts of the input can improve the computational efficiency of their streaming algorithm for k-Median.

Non-Metric Facility Location. Alon, Awerbuch, Azar, Buchbinder, and Naor [1] studied the *non-metric* version of Online Facility Location, where the distances need not satisfy the triangle inequality. Building on their work on Online Set Cover [2], Alon et al. presented a randomized $\mathrm{O}(\log n \log m)$-competitive algorithm for non-metric Facility Location, where n is the number of demands and m is the number of potential facility locations. The algorithm employs an elegant primal-dual algorithm which is $\mathrm{O}(\log m)$-competitive for the fractional version of the problem, where the algorithm can open a facility to any extent between 0 and 1, and a randomized rounding procedure, which is applied in an online fashion and increases the competitive ratio by a factor of $\mathrm{O}(\log n)$. On the negative side, the non-metric version of Online Facility Location is a generalization of Online Set Cover. Therefore, the lower bound of $\Omega(\frac{\log m \log n}{\log \log n + \log \log m})$ on the competitive ratio of any deterministic algorithm for Online Set Cover proven in [2] also applies to the non-metric version of Online Facility Location.

Facility Leasing. Nagarajan and Williamson [35] studied the problem of Facility Leasing, an interesting generalization of Facility Location introduced by Anthony and Gupta [4]. Facility Leasing is motivated by practical applications where the demands and the facilities serving them have a limited lifespan, rather than being permanent, and the facility costs obey economies of scale and are given by

a non-decreasing concave function of their lifespan. Thus, in Facility Leasing, the set of demands may change over time and the facility costs depend on the time length for which the facilities stay open.

More precisely, the input consists of a set of demands D_t which are active on each day t, and K possible lease types for each potential facility location. Each lease type determines the lease cost and the time length for which the facility stays open. Namely, for each lease type k, a facility z can be leased at any day t for l_k days at a cost of f_z^k, i.e., the facility z is open from day t until day $t + l_k - 1$, and any demand u active on some day τ, $t \leq \tau \leq t + l_k - 1$, can be assigned to z at a cost of $d(z, u)$. The goal is to minimize the total cost of facility leases plus the total cost of assigning the demands in each set D_t to some facility open at day t.

For the offline version of Facility Leasing, Nagarajan and Williamson proved that an elegant modification of the primal-dual algorithm of Jain and Vazirani [27] gives an approximation ratio of 3.

In the online version of Facility Leasing, the potential facility locations and their lease types are given to the algorithm in advance. In each day t, the demands in D_t arrive online and must be assigned to an open facility upon arrival. In fact, Online Facility Leasing is an interesting generalization of both Online Facility Location and Online Parking Permit [34]. Nagarajan and Williamson [35] presented a primal-dual $\mathrm{O}(K \log n)$-competitive algorithm for Online Facility Leasing. At the conceptual level, the algorithm of [35] combines PD-OFL with Meyerson's $\mathrm{O}(K)$-competitive deterministic algorithm for Parking Permit [34].

An interesting problem left open by [35] is to determine the competitive ratio of deterministic and randomized algorithms for Online Facility Leasing. The best known lower bounds are $\Omega(K + \frac{\log n}{\log \log n})$ and $\Omega(\log K + \frac{\log n}{\log \log n})$ respectively, and follow from the lower bounds on the competitive ratio of Online Facility Location [19] and Online Parking Permit [34]. Hence, it would be interesting to know whether the competitive ratio of Online Facility Leasing is $\Omega(K \frac{\log n}{\log \log n})$ for deterministic and $\mathrm{O}(\log K \frac{\log n}{\log \log n})$ for randomized algorithms, or there are online algorithms with a considerably better competitive ratio, close to the best known lower bounds. In simple words, we ask whether the spatial and the temporal dimensions of the problem are really independent from each other, or one can somehow combine the information collected for each of them and improve the competitive ratio.

Strategyproof Mechanisms for Facility Location Games. In the field of Algorithmic Game Theory, there has been a significant recent interest in approximate strategyproof and group strategyproof mechanisms for Facility Location Games, where a number of facilities are placed in a metric space based on the preferences of strategic agents (see e.g., [37, 30, 29, 20]). In a Facility Location game, n agents report their locations to a mechanism, which based on them, determines a set of facilities to open. The agents are selfish, and seek to minimize their own assignment cost, namely the distance of their (true) location to the nearest facility. In fact, an agent may even falsely report her location in an attempt of manipulating the mechanism. The mechanism cannot give any monetary incentives to the agents (or compensate them in any other way), and should be *strategyproof*, i.e. should ensure that no agent can benefit from misreporting her location, or even *group strategyproof*, i.e. should ensure that for any group of agents misreporting their locations, at least one of them does not benefit. At the same time, the mechanism should optimize, or at least approximate, some reasonable notion of social cost.

Recent work on strategyproof mechanisms for Facility Location games adopts k-Median as the prevailing notion of social cost. Namely, the mechanism seeks to minimize the assignment cost for all agents, given k facilities to open. However, known impossibility results suggest that even in some very simple settings, such as 1-Median on non-tree metrics and 2-Median on the line, the optimal solution is not strategyproof (see e.g., [38, 37]). Therefore, recent work mostly focuses on the approximability of some simple special cases of k-Median, such as 1-Median on general metrics and 2-Median on the line, by deterministic and randomized strategyproof mechanisms (see e.g., [37, 30, 29]).

Motivated by the absence of any positive or negative results if the mechanism may open more than 2 facilities, [20] investigated the approximability of Facility Location by deterministic and randomized (group) strategyproof mechanisms. Among other results, [20] proved that a variant of $\mathrm{PartOFL}$ is group strategyproof and achieves an approximation ratio of $\mathrm{O}(\log n)$ for Facility Location on the real line. Moreover, [20] introduced the notion of *winner-imposing* mechanisms, which require that if a facility is placed at an agent's reported location, the agent should connect to it, and proved that the winner-imposing version of $\mathrm{RandOFL}$ is strategyproof. Therefore, applying $\mathrm{RandOFL}$ to a random permutation of the agents' locations, [20] obtained an 8-approximate randomized winner-imposing strategyproof mechanism for Facility Location on general metric spaces. A really interesting point is that the results of [20] may suggest a more general connection between approximate mechanism design without monetary transfers and online optimization.

References

1. N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A General Approach to Online Network Optimization Problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.
2. N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. The Online Set Cover Problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.
3. A. Anagnostopoulos, R. Bent, E. Upfal, and P. Van Hentenryck. A Simple and Deterministic Competitive Algorithm for Online Facility Location. *Information and Computation*, 194:175–202, 2004.
4. B.M. Anthony and A. Gupta. Infrastructure Leasing Problems. In *Proc. of the 12th Conference on Integer Programming and Combinatorial Optimization (IPCO '07)*, LNCS 4513, pp. 424–438, 2007.
5. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local Search Heuristics for k-Median and Facility Location Problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
6. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
7. J. Byrka and K. Aardal. An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010.
8. M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental Clustering and Dynamic Information Retrieval. In *Proc. of the 29th ACM Symposium on Theory of Computing (STOC '97)*, pp. 626–635, 1997.
9. M. Charikar and S. Guha. Improved Combinatorial Algorithms for Facility Location Problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.
10. M. Charikar, L. O'Callaghan, and R. Panigrahy. Better Streaming Algorithms for Clustering Problems. In *Proc. of the 35th ACM Symposium on Theory of Computing (STOC '03)*, pp. 30–39, 2003.
11. M. Charikar and R. Panigrahy. Clustering to Minimize the Sum of Cluster Diameters. In *Proc. of the 33rd ACM Symposium on Theory of Computing (STOC '01)*, pp. 1–10, 2001.
12. M. Chrobak and C. Kenyon-Mathieu. Online Algorithms Column 10: Competitiveness via Doubling. *SIGACT News*, 37(4):115–126, 2006.
13. G. Divéki and C. Imreh. Online Facility Location with Facility Movements. *Central European Journal of Operations Research*, 2010.
14. Z. Drezner and H.W. Hamacher (Editors). *Facility Location: Applications and Theory*. Springer, 2004.
15. J. Fakcharoenphol, S. Rao, and K. Talwar. Approximating Metric Spaces by Tree Metrics. *Encyclopedia of Algorithms*, pp. 51–53, 2008.
16. D. Fotakis. Incremental Algorithms for Facility Location and k-Median. *Theoretical Computer Science*, 361(2-3):275–313, 2006.
17. D. Fotakis. Memoryless Facility Location in One Pass. In *Proc. of the 23st Symposium on Theoretical Aspects of Computer Science (STACS '06)*, LNCS 3884, pp. 608–620, 2006.
18. D. Fotakis. A Primal-Dual Algorithm for Online Non-Uniform Facility Location. *Journal of Discrete Algorithms*, 5(1):141–148, 2007.
19. D. Fotakis. On the Competitive Ratio for Online Facility Location. *Algorithmica*, 50(1):1–57, 2008.
20. D. Fotakis and C. Tzamos. Winner-Imposing Strategyproof Mechanisms for Multiple Facility Location Games. In *Proc. of the 6th Workshop on Internet and Network Economics (WINE '10)*, LNCS 6484, pp. 234–245, 2010.
21. G. Frahling and C. Sohler. Coresets in Dynamic Geometric Data Streams. In *Proc. of the 37th ACM Symposium on Theory of Computing (STOC '05)*, pp. 209–217, 2005.
22. S. Guha and S. Khuller. Greedy Strikes Back: Improved Facility Location Algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

23. S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528, 2003.
24. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering Data Streams. In *Proc. of the 41st IEEE Symposium on Foundations of Computer Science (FOCS '00)*, pp. 359–366, 2000.
25. P. Indyk. Algorithms for Dynamic Geometric Problems over Data Streams. In *Proc. of the 36th ACM Symposium on Theory of Computing (STOC '04)*, pp. 373–380, 2004.
26. K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy Facility Location Algorithms Analyzed Using Dual Fitting with Factor-Revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.
27. K. Jain and 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.
28. C. Lammersen and C. Sohler. Facility Location in Dynamic Geometric Data Streams. In *Proc. of the 16th European Symposium on Algorithms (ESA '08)*, LNCS 5193, pp. 660–671, 2008.
29. P. Lu, X. Sun, Y. Wang, and Z.A. Zhu. Asymptotically Optimal Strategy-Proof Mechanisms for Two-Facility Games. In *Proc. of the 11th ACM Conference on Electronic Commerce (EC '10)*, pp. 315–324, 2010.
30. P. Lu, Y. Wang, and Y. Zhou. Tighter Bounds for Facility Games. In *Proc. of the 5th Workshop on Internet and Network Economics (WINE '09)*, LNCS 5929, pp. 137–148, 2009.
31. M. Mahdian, Y. Ye, and J. Zhang. Improved Approximation Algorithms for Metric Facility Location Problems. In *Proc. of the 5th Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX '02)*, LNCS 2462, pp. 229–242, 2002.
32. R.R. Mettu and C.G. Plaxton. The Online Median Problem. *SIAM Journal on Computing*, 32(3):816–832, 2003.
33. A. Meyerson. Online Facility Location. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS '01)*, pp. 426–431, 2001.
34. A. Meyerson. The Parking Permit Problem. In *Proc. of the 46th IEEE Symposium on Foundations of Computer Science (FOCS '05)*, pp. 274–284, 2005.
35. C. Nagarajan and D.P. Williamson. Offline and Online Facility Leasing. In *Proc. of the 13th Conference on Integer Programming and Combinatorial Optimization (IPCO '08)*, LNCS 5035, pp. 303–315, 2008.
36. P.B. Mirchandani and R.L. Francis (Editors). *Discrete Location Theory*. Willey, 1990.
37. A.D. Procaccia and M. Tennenholtz. Approximate Mechanism Design Without Money. In *Proc. of the 10th ACM Conference on Electronic Commerce (EC '09)*, pp. 177–186, 2009.
38. J. Schummer and R.V. Vohra. Strategyproof Location on a Network. *Journal of Economic Theory*, 104:405–428, 2002.
39. D. Shmoys. Approximation Algorithms for Facility Location Problems. In *Proc. of the 3rd Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX '00)*, LNCS 1913, pp. 27–33, 2000.
40. D. Shmoys, E. Tardos, and K. Aardal. Approximation Algorithms for Facility Location Problems. In *Proc. of the 29th ACM Symposium on Theory of Computing (STOC '97)*, pp. 265–274, 1997.
41. M. Sviridenko. An Improved Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. In *Proc. of the 9th Conference on Integer Programming and Combinatorial Optimization (IPCO '02)*, LNCS 2337, pp. 240–257, 2002.
42. M. Thorup. Quick k-Median, k-Center, and Facility Location for Sparse Graphs. *SIAM Journal on Computing*, 34(2):405–432, 2005.