On the Power of Priority Algorithms for Facility Location and Set Cover

Spyros Angelopoulos and Allan Borodin
Department of Computer Science
University of Toronto
Toronto, ON, Canada M5S 3G4
{spyros,bor}@cs.toronto.edu

May 28, 2003

Abstract

We apply and extend the priority algorithm framework introduced by Borodin, Nielsen and Rackoff to define "greedy-like" algorithms for (uncapacitated) facility location problems and set cover. These problems have been the focus of extensive research from the point of view of approximation algorithms, and for both problems, greedy-like algorithms have been proposed and analyzed. The priority algorithm definitions are general enough so as to capture a broad class of algorithms that can be characterized as "greedy-like" while still possible to derive non-trivial lower bounds on the approximability of the problems by algorithms in such a class. Our results are orthogonal to complexity considerations, and hence apply to algorithms that are not necessarily polynomial-time.

1 Introduction

Under the assumption that $P \neq NP$, NP-hard optimization problems do not admit polynomial-time optimal algorithms. The natural question "how well can we approximate an NP-hard problem by a polynomial-time algorithm" has been the motivation for extensive research over the last decade, and the hardness of approximating a problem has developed into a a very prominent field within the area of approximation algorithms. On the other hand, very little is known about the approximation power of *specific* computational paradigms. For instance, what can we say about the hardness of approximating a given optimization problem by an algorithm that we would intuitively characterize as "greedy-like"?

The latter question was addressed by Borodin, Nielsen and Rackoff [5] who introduced the class of so-called *priority algorithms* as a model for describing natural greedy-like algorithms. They also proposed a framework for proving lower bounds on the approximability of a problem by such a class of algorithms. To illustrate the techniques, they applied the framework to a number of well-known scheduling problems, but claimed that the framework could become applicable to other problems as well.

In this paper, we follow the priority-algorithm framework in [5] so as to to characterize "greedy and greedy-like algorithms" for the uncapacitated facility location problem and the set cover problem. These well-studied and related NP-hard problems are central problems in the study of approximation algorithms. (See section 3 for further discussion of some of the relevant results.) The best known polynomial time computable approximation ratio (essentially $\ln n$, where n is the number of elements in the underlying universe) for the (weighted) set cover problem is achieved by a most natural greedy algorithm [13] [15] [6] and quite good approximation ratios (namely, 1.61 [12] and 1.861 [16]) have been derived by greedy-like algorithms for the metric uncapacitated facility location problem. For these two optimization problems, we apply the priority-algorithm framework and derive lower bounds on the approximation ratio for algorithms in this class. Informally, priority algorithms are characterized by the following two properties:

1. The algorithm specifies an ordering of "the input items" and each input item is considered in this order.

2. As each input item is considered, the algorithm must make an "irrevocable decision" concerning the input input.

To make these concepts precise, (i.e., what precisely an "input item" and an "irrevocable decision" are) one has to apply the framework to particular problems (see section 2.2). The goal is to make the definitions sufficiently general and robust so as to capture known algorithms and anything that would be classified as "greedy-like" while still being sufficiently restrictive that interesting lower bounds can be derived. As in the competitive analysis of online algorithms, priority algorithm lower bounds are orthogonal to complexity bounded approximation ratios. That is, although greedy algorithms are highly desirable because they tend to be time efficient, the definition of priority algorithms permits arbitrarily complex (in terms of computing time) computation while deriving lower bounds by exploiting the structure of the algorithm.

Depending on whether the ordering changes throughout the execution of the algorithm, two classes of priority algorithms can be defined:

- Algorithms in the class Fixed Priority decide the ordering before any input item is considered and this ordering does not change throughout the execution of the algorithm.

- Algorithms in the more general class Adaptive Priority are allowed to specify a new ordering after each input item is processed. The new ordering can thus depend on input items already considered.

As argued in [5], a further distinction can be made for Adaptive Priority algorithms, depending on the information the algorithm maintains on input items considered in the past. At one extreme, the irrevocable decision in a *memoryless* Adaptive Priority algorithm depends only on the "current configuration"; (e.g., for the facility location problem, only on facilities already considered *and opened* in the solution being constructed). At the other extreme, the algorithm can keep track of all input items considered in the past, and the corresponding decisions (e.g., in facility location, the algorithm knows whether a facility is has already considered has been opened).

As in [5], "greedy algorithms" are (fixed or adaptive) priority algorithms, which satisfy an additional property: the irrevocable decision is such that the objective function is *locally optimized.* More specifically, the objective function must be optimized as if the the input currently being considered is the last input. The definition suggests that it is the nature of the decision and not the ordering that determines "greediness".

Within this framework, we prove the following lower bounds:

1. The set cover problem over a universe of n elements.

(a) No ADAPTIVE PRIORITY algorithm can achieve a better approximation ratio than the precise bound $(\ln n - \ln\ln n + \Theta(1))$ obtained by Slavík [22] for *the* greedy algorithm (see section 3). This lower bound applies to the *uniform* set cover problem in which all set costs are the same. Note that Feige [7] shows that under a reasonable complexity assumption (i.e., assuming that NP is not contained in $DTIME(n^{O(\log\log n)})$), it is not possible to obtain a polynomial-time approximation of $(1-\epsilon)\ln n$ for any $\epsilon > 0$.

(b) For any $\epsilon > 0$, no FIXED PRIORITY algorithm can achieve an approximation ratio better than $(1-\epsilon)n$.

Since the set cover problem can be viewed as a special case of the facility location problem (with distances in $\{0, \infty\}$), all lower bounds for the set cover problem hold for the facility location problem when arbitrary distances are allowed.

2. The metric uncapacitated facility location problem. All of the following lower bounds will apply to the unweighted case (i.e., each city has weight 1), with facilities having *uniform* (i.e., identical) opening costs, and where all distances are in $\{1, 3\}$. The best known corresponding greedy upper bounds apply to the weighted case and for an arbitrary metric distance function.

   (a) No ADAPTIVE PRIORITY algorithm with unbounded memory can achieve an approximation ratio better than 4/3.

   (b) For memoryless ADAPTIVE PRIORITY algorithms we show a lower bound of 1.463 on the approximation ratio of the metric facility location problem, which matches the complexity-based bound of Guha and Khuller [8] (under the same assumption used by Feige [7]).

   (c) For all $\epsilon > 0$, no FIXED PRIORITY GREEDY algorithm can achieve an approximation ratio of $3 - \epsilon$ for the uniform metric facility location problem. This bound matches the 3-approximation upper bound of Mettu and Plaxton [18] which applies to the non-uniform case. For FIXED PRIORITY but not necessarily GREEDY algorithms, we show a corresponding bound of 1.366 (with no memory restrictions).

It is interesting to note that certain primal-dual algorithms can be seen as priority algorithms, or, more precisely, certain primal-dual algorithms have an alternative combinatorial statement as a priority algorithm. For instance, the greedy algorithm for set cover [13] [15] [6] and the greedy-like algorithms for facility location due to Mahdian *et al* [16] and Jain, Mahdian and Saberi [12] can be stated as primal-dual algorithms. Of course, not every primal-dual algorithm is a priority algorithm; for example, primal-dual algorithms that apply a reverse-delete (or, generally speaking, reverse-reconstruction) step do not appear to belong in the class of priority algorithms. Hence, a negative result concerning a priority algorithm suggests that a reverse-reconstruction phase is needed if one wants to design a primal-dual algorithm with better performance.

Naturally, defining what precisely constitutes a "greedy-like" algorithm is very subjective. As observed in [5], it is unlikely that priority algorithms can capture every possible algorithm that intuition would classify as "greedy-like"; for instance, the one-pass *Admission Algorithm* of Bar-Noy, Guha, Naor and Schieber [4] for a general version of a scheduling problem, requires that certain decisions (in particular, the admission of a job in the schedule) are not irrevocable, but only temporarily, and as such they can be revoked in the future. In other words, the above algorithm applies some limited backtracking, which may or may not be a characteristic of a "greedy-like" algorithm. On the other hand, a greedy-like algorithm may be allowed to maintain global information that the priority-algorithm framework (or every natural model, for that matter) does not or cannot model. Nevertheless, the framework does include well-known algorithms that have, undoubtedly, greedy-like flavor (see section 3 for examples of such algorithms), and provides a solid starting point towards a classification based on "greediness".

2 Preliminaries

2.1 Problem statements

In the *(uncapacitated, unweighted) facility location* problem, the input consists of a set $\mathcal{F}$ of facilities and a set $\mathcal{C}$ of cities with $\mathcal{F} \cap \mathcal{C} = \emptyset$ [1]. Each facility $i \in \mathcal{F}$ is associated with an *opening cost* f_i which reflects the cost that must be paid to utilize the facility. Furthermore, for every facility $i \in \mathcal{F}$ and city $j \in \mathcal{C}$, the non-negative *distance* or *connection cost* c_{ij} is the cost that must be paid to connect city j to facility i. The objective is to open a subset of the facilities in $\mathcal{F}$ and connect each city in $\mathcal{C}$ to an open facility so that the total cost incurred, namely the sum of the opening costs and the connection costs, is minimized. In the *metric* version of the problem, the connection costs satisfy the triangle inequality.

We emphasize that we assume the model in which facilities and cities are *disjoint* sets. Even though this assumption is not needed from the point of view of upper bounds (algorithms), the arguments we will present rely strongly on it; in particular, the lower bounds we present do not hold in the *complete model*, in which every node is both a facility and a city.

In the *set cover* problem, we are given a universe U of n elements, and a collection $\mathcal{S}$ of subsets of U. Each set $S \in \mathcal{S}$ is associated with a cost $c(S)$. We seek a minimum-cost subcollection of $\mathcal{S}$ that covers all elements of U; that is, a collection of sets $\mathcal{S}' \subseteq \mathcal{S}$ of minimum total cost such that for every $e \in U$ there exists a set $S \in \mathcal{S}'$ with $e \in S$.

2.2 Definitions of input items for priority algorithms

What precisely constitutes the input for the problems we study? For the uncapacitated facility location problem the cost of a solution is determined by the set of facilities the algorithm decides to open, since each city will be connected to the nearest open facility. Hence, we will assume that the input items are the facilities themselves, where each facility is identified by a unique id, its distance to every city and its opening cost. When considering a facility (of highest priority), the algorithm must make an irrevocable decision as to whether or not to open this facility. The decision is irrevocable, in the sense that once a facility is opened, its opening cost will count towards the total cost that the algorithm incurs and if a facility is not opened when considered, it cannot be opened at a later stage.

For the class of scheduling problems considered in [5], there is no issue as to what the "input items" are. But for the facility location problem (and the set cover problem), there is at least one other very natural way to view the input items. Namely, as in Meyerson [19], we can think of the cities as being the input items, with each city being identified by its id, and its distance to each facility. (We can treat the opening costs of all facilities as global information.) The irrevocable decision would then be to assign each input city to some open facility, possibly opening a new facility if desired [2]. Indeed this model is very natural in the sense that one would expect the

[1] We assume that $\mathcal{F} \cap \mathcal{C} = \emptyset$ noting that if $x \in \mathcal{F} \cap \mathcal{C}$, then we can replace x by $x_f \in \mathcal{F}$ and $x_c \in \mathcal{C}$ with zero connection cost between x_f and x_c.

[2] Meyerson's algorithms apply to the complete model, as defined earlier in the section.

number of cities to be much larger than the number of facilities. We have chosen our model as it abstracts several known $O(1)$-approximation greedy-like algorithms (see section 3). We note, however, that it is possible to show negative results for priority algorithms in the model in which the input items are the cities [1].

Similar to the facility location problem, there are two natural ways to represent the input for the set cover problem. Perhaps the most natural representation is to think of the input as consisting of sets, where each set is identified by an id, by its cost, and by the elements it covers. For such a model, we impose the constraint that the priority algorithm must irrevocably select any set it considers in case the set covers a new element, that is, an element not covered by the sets already selected. This requirement is motivated by the observation that if the set that covers the new element is the only set with this property, and the algorithm does not select it, the resulting solution will not be feasible. In other words, the priority algorithm is oblivious of the sets to be considered in the future, and cannot anticipate whether feasibility will be upheld by not selecting a particular set. The well-known greedy algorithm for set cover [13] [15] observes the above requirement. It is worth mentioning that in this context every priority set cover algorithm belongs in the class of GREEDY algorithms [3]. Alternatively, it is possible to represent the input as a set of elements, where each element is described by a unique id and a collection of sets it is contained in; in this representation, we can treat set costs as global information. For reasons similar to the ones given for the facility location problem, in this paper we will consider the representation of the input as a collection of sets, rather than elements.

2.3 Orderings and the role of the adversary

To show a lower bound on the approximation ratio we must evaluate the performance of every priority algorithm for an appropriately constructed nemesis input. The construction of such a nemesis input can be seen as a game between an adversary and the algorithm and is conceptually similar to the construction of an adversarial input in competitive analysis. However, in the setting of priority algorithms, it is the algorithm and not the adversary that chooses the ordering of the inputs. But we do not want the algorithm to be able to chose an optimal ordering for each input (e.g. chose the optimal set of facilities as those of highest priority). One possibility is to define the "allowable orderings" to be those that are induced by functions mapping the set of all input items into the non negative reals. The nature of the adversary described below provides a more inclusive definition for what orderings are allowed.

We consider the basic framework where the priority algorithm has no additional global information (such as the total number of input items [4] ,the total "cost" of all items in the input, etc.) beyond the input itself. In this setting, the game between the adversary and a FIXED PRIORITY algorithm can be described as follows. The adversary presents a large set of potential input items S. The algorithm determines a total ordering on this set. The adversary then selects a subset S' of S as the actual input; that is, the adversary discards part of the potential input. We enforce the condition that the removal of input items does not affect the priority of the remaining input items. As an example of this, in our framework of a priority algorithm for facility location, a facility is not defined in terms of other facilities; therefore, when the adversary removes a facility, the priority of every other input item is unaffected[5].

[3] In contrast, every solution for an instance of facility location in which at least one facility is open, is feasible. Thus, not every priority facility location algorithm is necessarily greedy.

[4] For the problems considered in this paper, it is not hard to show that we can extend our lower bounds to the case that the algorithm does know the number of input items.

[5] This explains why we need to restrict our attention to inputs in which the set of facilities and the set of cities

The situation is not much different for the class Adaptive Priority. Even though the algorithm can determine a new ordering after considering each input item, the adversary can adaptively remove input items from S (in each iteration) so as to derive the actual input S'.

Following the discussion in [5], we define *memoryless* (adaptive) facility location algorithms. In general, an adaptive algorithm defines a new ordering at every iteration based on all the input items (and hence the corresponding decisions) thus far considered. In contrast, the ordering of a memoryless algorithm depends only on the current configuration. That is, in the context of the facility location problem, the ordering in any iteration depends only on the set of facilities that have been opened and not on any facilities that have been considered but not opened.

3 Related work

Facility location problems have been the focus of extensive research from the operations research and the computer science communities for several decades. However, it was only recently that Shmoys, Tardos and Aardal [21] gave the first $O(1)$ polynomial time approximation algorithm. The past several years have witnessed a steady series of improvements on the approximability of metric facility location. The approaches used include LP-rounding, the primal-dual method, local search, dual fitting, and combinations of the above. We refer the interested reader to the survey of Shmoys [20], for a collection of recent developments in the area. Currently, the best-known approximation ratio (1.52) for metric facility location is due to Mahdian, Ye and Zhang [17], and is achieved by a non-priority algorithm. For the special case of connection costs in $\{1,3\}$, the LP-based (non-priority) algorithm of Guha and Khuller [8] provides a 1.463-approximation, matching the complexity based hardness result presented in the same paper.

We identify algorithms that follow the paradigm of a priority algorithm, as defined in section 2.2. Hochbaum [10] presented an algorithm in the class Adaptive Priority Greedy which is an $O(\log n)$ approximation for facility location in arbitrary spaces, and showed that the analysis for the particular algorithm is tight. Mahdian, Markakis, Saberi and Vazirani [16] showed that a natural Adaptive Priority algorithm, which is only a small modification of Hochbaum's algorithm, is a 1.861 approximation algorithm for metric facility location; the analysis is performed by an elegant application of the dual-fitting technique. In addition, this algorithm can be made into a memoryless one with the same approximation ratio. To our knowledge, the best approximation achieved by a priority algorithm is due to Jain, Mahdian and Saberi [12]: their algorithm belongs in the class Adaptive Priority as well, and it is also analyzed by using the dual fitting technique. Mettu and Plaxton [18] showed that an algorithm which belongs in the class Fixed Priority Greedy yields a 3-approximation for metric facility location in the complete model.

Set cover is one of the oldest and most well-studied NP-hard problems. Johnson [13] and Lovász [15] proposed a simple greedy algorithm which they showed provides a $H(n)$-approximation, where $H(n)$ is the n-th harmonic number, and $n = |U|$. Chvátal [6] extended their results to the weighted case. A tight analysis of the greedy algorithm due to Slavík [22] is of particular interest, as discussed in section 4.2. This specific greedy algorithm iteratively selects the most cost-effective set, i.e., the set that minimizes the average cost at which it covers new (i.e., currently uncovered) elements. Clearly, the above algorithm belongs in the class Adaptive Priority: in every iteration, the priority of a set is its cost-effectiveness. In this paper, we refer to this algorithm as *the* greedy algorithm for set cover, to distinguish it from other (greedy) priority algorithms that one can define.

In terms of negative results, Feige [7] showed that it is not possible to obtain a polynomial-

are disjoint, as mentioned earlier.

time approximation of $(1-\epsilon)\ln n$ for set cover, unless NP has slightly superpolynomial time deterministic algorithms. On a similar vein, Arora and Sudan [3] showed that set cover is not approximable within a factor of $o(\log n)$, under the weaker assumption that $P \neq NP$.

It must be mentioned that the priority-algorithm framework of Borodin, Nielsen and Rackoff [5] does not seem to be directly applicable to graph optimization problems without certain modifications. In recent work, Impagliazzo and Davis [11] addressed this issue and proposed an extension of the framework so as to capture greedy-like algorithms for this class of problems.

As one expects, the question "how well can a specific class of algorithms approximate a problem" is not (or should not be) restricted to greedy-like algorithms only. Motivated by the lack of IP formulations for the vertex cover problem having integrality gap better than $2-\epsilon$, Arora, Bolobás and Lovász [2] recently showed that this bound is tight for three broad classes of IP formulations for this problem. The results suggest that every linear relaxations for any IP formulation that falls in one these classes will not yield any improvement on the approximation ratio of vertex cover. In a similar flavor, Khanna, Motwani, Sudan and Vazirani [14] proved lower bounds for the so-called class of *oblivious* local-search algorithms for problems such as MAX 2-SAT and MAX 2-CSP. The above class contains local search algorithms which apply a greedy rule in the choice of the local change. Finally, another limited computational paradigm that is close to priority algorithms is *data stream algorithms* (see, e.g., [9]). Here, the algorithm is allowed to perform either a single pass, or only a small number of passes over the data, and is also subject to constraints on the amount of memory it can use. Both upper and lower bounds for algorithms in this class have been shown.

4 Adaptive priority algorithms

In this section we show lower bounds on the approximability of set cover and facility location by adaptive priority algorithms. All our lower-bound constructions for metric facility location use connection costs in $\{1,3\}$, suggesting the following definitions. Let C_f be the set of cities at distance 1 from facility f. We say that f *covers* C_f. The *complement of* f *with respect to* C where $C_f \subseteq C$ is defined as the facility that covers all and only the cities in $C \setminus C_f$, and has the same facility opening cost as f. For simplicity, when C is the set $\mathcal{C}$ of all cities, we say that f and $\overline{f}$ are *complementary*, where $\overline{f}$ is the complement of f wrt $\mathcal{C}$.

4.1 Adaptive priority facility location with unbounded memory

We first consider the most general case of an Adaptive Priority algorithm with unbounded memory (i.e., the algorithm has complete knowledge of the decisions made in the past). In section 4.3 we show how to derive improved bounds for priority algorithms which are memoryless.

Theorem 1 *No priority algorithm for the uniform metric facility location problem can achieve an approximation ratio better than* $\frac{4}{3}$.

Proof. Define an instance of the uniform metric facility location problem as follows. The set of cities $\mathcal{C}$ consists of n cities. Each facility is identified with the set of cities it covers. More precisely, for every $C \subset \mathcal{C}$ with $|C| = \frac{n}{2}$, there exists a facility f that covers C (and only cities in C) [6]. For every city in $\mathcal{C} \setminus C$, the cost of connecting the city to f is equal to 3. Note that for every facility f in the instance, the complement $\overline{f}$ of f is also in the instance.

[6] Throughout this paper we ignore floors and ceilings and assume that n is appropriately divisible.

Every time the algorithm considers a facility f it must decide whether to open it. If it opens f, the adversary removes all facilities which do not cover exactly $\tilde{n}/2$ uncovered cities, where $\tilde{n}$ is the number of currently uncovered cities. The adversary also removes $\overline{f}$ unless $\overline{f}$ is the last remaining facility in the input set. If the algorithm does not open f, the adversary removes all remaining facilities except for $\overline{f}$. Let all facility costs be equal to $\frac{n}{4}$. The optimal algorithm will open a pair of complementary facilities at an opening cost of $2 \cdot \frac{n}{4}$, and connect all cities at a cost of n, for a total cost of $\frac{3}{2}n$. Suppose that the algorithm opens k facilities, with $k \geq 1$. It is easy to see that the union of these k facilities covers at most $n \sum_{i=1}^{k} 2^{-i} = n(1 - 2^{-k})$ cities. Hence the algorithm pays a facility-opening cost of $k \cdot \frac{n}{4}$ and a connection cost of at least $n(1-2^{-k}) + 3n \cdot 2^{-k}$, and thus a total cost of $(1 + \frac{k}{4} + 2^{-k+1})n$. This expression is minimized at either $k = 2$ or $k = 3$, giving a $\frac{4}{3}$ ratio between the algorithm's cost and the optimal cost. □

Using an argument along the same lines one can prove the following:

Theorem 2 *The approximation ratio of every priority algorithm for facility location in arbitrary spaces is* $\Omega(\log n)$.

Proof. We consider the same set of cities C_f covered by a facility f except now any city not covered by f (i.e. not at distance 1) has ∞ distance from f. In this case, any priority algorithm must open any facility it considers. Hence it will consider and open $\log n$ facilities while the optimal algorithm will again open two facilities. □

Note that the bound is tight, since Hochbaum [10] showed that a simple greedy-like algorithm, which can be classified as Adaptive Priority, is an $O(\log n)$ approximation algorithm for the problem.

4.2 Adaptive priority set cover

In this section we prove that no (adaptive) priority algorithm can perform better than *the* greedy set-cover algorithm.

A remarkably tight analysis of the greedy set cover algorithm due to Slavík [22] shows that its approximation ratio is exactly $\ln n - \ln \ln n + \Theta(1)$, where $n = |U|$ is the size of the universe. For the proof of the lower (as well as the upper) bound, Slavík considered an instance of set cover with sets of uniform cost on a universe U of $n \geq N(k, l)$ elements. Here, $N(k, l)$ is defined as the smallest size of U such that there is a set cover instance over U with the property that the cost of the greedy algorithm is exactly k, while the cost of the optimal algorithm is exactly l, for given k, l with $k \geq l$. The collection of input sets for this specific instance consists of two subcollections, each covering all n elements of U. The first subcollection, denoted by $\mathcal{S}_1$, contains $n_1, n_2, \ldots, n_r$ disjoint sets of sizes $s_1, s_2, \ldots, s_r$, respectively, so that $\sum_{i=1}^{r} n_i s_i = n$. The numbers n_i's and sizes s_i's of the sets are selected in [22] appropriately, but for the purposes of our discussion it suffices to mention the following: i) $n_i > 0$, for all i with $1 \leq i \leq r$, and r is a postitive integer; ii) $\sum_{i=1}^{r} n_i = k$; iii) $s_1 > s_2 > \ldots > s_r > 0$; and iv) $s_1 \geq \lceil \frac{n}{l} \rceil$. The second subcollection, denoted by $\mathcal{S}_2$, consists of l sets of sizes in $\{\lceil \frac{n}{l} \rceil, \lceil \frac{n}{l} \rceil - 1\}$. Slavík argued that the greedy algorithm must select all sets in $\mathcal{S}_1$, at a total cost of $\sum_{i=1}^{r} n_i$. In addition, the greedy algorithm selects the sets in $\mathcal{S}_1$ in non-increasing order of sizes, breaking ties arbitrarily. On the other hand, the cost of the optimal algorithm is the total cost of the subcollection $\mathcal{S}_2$, namely, l. Slavík showed that for every n sufficiently large, one can find k, l such that $N(k, l) \leq n < N(k+1, l)$, and $\frac{k}{l} \geq \ln n - \ln \ln n + \Theta(1)$. We will show that, for every such n (and then k and l chosen as in Slavík's analysis), there exists an instance of set cover with $|U| = n$ for which the cost of every priority algorithm is at least $k = \sum_{i=1}^{r} n_i$, while the cost of the optimal algorithm is at most l. Then Slavík's analysis carries over to yield the same lower bound on the approximation ratio for every priority algorithm.

Consider an instance of set cover which consists of a universe U of size n and all $\binom{n}{d}$ sets of size d, where $d = s_1 \geq \lceil \frac{n}{l} \rceil$. All sets have identical costs. Given an adaptive priority algorithm for the problem, we describe the actions of the adversary, which takes place in r phases. As we will show in Lemma 3, it is feasible to define such an adversary. Phase 1 begins with the first set selected by the algorithm; it terminates when the algorithm has selected exactly n_1 sets that cover at least one new element each. If the number c_1 of elements covered by the algorithm in phase 1 is less than $n_1 s_1$, the adversary chooses any $n_1 s_1 - c_1$ uncovered elements, which together with the c_1 covered elements form a set of elements denoted by C_1. Without loss of generality, we will consider all elements in C_1 as being covered. The adversary removes all sets from the input except for sets that contain at least $d - s_2$ elements of C_1, namely a set S remains in the input only if $|S \cap C_1| \geq d - s_2$ (recall that $d = s_1 > s_2$). Phase 2 then begins. Likewise, phase i, with $1 \leq i \leq r - 1$ terminates when the algorithm has selected exactly n_i sets that cover at least one uncovered element each. If the number c_i of elements covered in phase i is less than $n_i s_i$, the adversary chooses any of the remaining $n_i s_i - c_i$ elements, which together with the c_i covered elements are the contents of a set denoted by C_i. (We again consider all elements in C_i as being covered.) The adversary then removes all input sets except for sets that contain at least $d - s_{i+1}$ elements of $\bigcup_{j=1}^{i} C_j$, namely, a set S remains in the input only if $|S \cap \bigcup_{j=1}^{i} C_j| \geq d - s_{i+1}$. The adversary does not remove any sets at the end of (the last) phase r. All sets not explicitly removed by the adversary comprise the actual input that is presented to the algorithm.

We need to show that it is feasible to define an adversary as above. Denote by R_i the collection of sets that remain in the input at the beginning of phase i, with $1 \leq i \leq r$. Note that every set in R_i can cover at most s_i new (so far uncovered) elements.

Lemma 3 *At the beginning of phase i, at least $n_i s_i$ elements are uncovered, and the sets in R_i can cover these elements.*

Proof. By induction on i. Recall that $\sum_{j=1}^{r} n_j s_j = n$. Clearly, at the beginning of phase 1 there are $n \geq n_1 s_1$ uncovered elements, and R_1 consists of all $\binom{n}{d}$ sets, hence the $n_1 s_1$ uncovered elements can be covered by sets in R_1. Consider the beginning of phase i, with $1 < i \leq r$. In each phase j, with $1 \leq j < i$, the algorithm covered $|C_j| = n_j s_j$ elements (by the definition of the adversary and the induction hypothesis). Therefore, at the beginning of phase i, $n - \sum_{j=1}^{i-1} n_j s_j = \sum_{j=i}^{r} n_j s_j \geq n_i s_i$ elements are uncovered. Let P be a set that contains exactly $n_i s_i$ uncovered elements. Let $P_1, P_2, \ldots, P_{n_i}$ be any disjoint partition of P in sets of size s_i. We claim that for any m with $1 \leq m \leq n_i$ there exists at least one set in R_i that covers P_m. Define the set S as the set that contains exactly $s_h - s_{h+1}$ elements of C_h, for every h with $1 \leq h < i$ and also contains P_m. Since the C_h's with $h < i$ and P_m are all disjoint, the set S has size

$$\sum_{h=1}^{i-1} (s_h - s_{h+1}) + s_i = (s_1 - s_i) + s_i = s_1 = d$$

hence S belongs in R_1. It remains to show that S is in R_i. To this end, note that for all $j < i$,

$$|S \cap \bigcup_{h=1}^{j} C_h| = \sum_{h=1}^{j} (s_h - s_{h+1}) = s_1 - s_{j+1},$$

therefore S is in R_{j+1}, for all $j < i$ (and in particular, for $j = i - 1$). Thus the adversary will not remove S before the end of phase i. $\square$

Denote by $cost(ALG)$, $cost(OPT)$ the cost of the priority algorithm and the cost of the optimal algorithm, respectively. We then have:

Lemma 4 $cost(ALG) \geq \sum_{i=1}^{r} n_i = k$.

Proof. This follows from Lemma 3. In particular, the algorithm will always select n_i sets in phase i, and thus will move to phase $i+1$, for all $i < r$. $\square$

The following combinatorial lemma will be useful in upper-bounding the optimal cost.

Lemma 5 *Suppose that there exists a collection of (not necessarily disjoint) sets $P_1, P_2, \ldots, P_l \subseteq U$, each of size d, with the following properties:*

- *$P_1 \ldots P_l$ cover all elements of U, namely $\bigcup_{h=1}^{l} P_h = U$.*
- *There exist disjoint sets $G_1, G_2, \ldots, G_r \subseteq U$, with $|G_j| = n_j s_j$, such that $|P_h \cap \bigcup_{j=1}^{i} G_j| \geq d - s_{i+1}$, for all $i \leq r-1$ and all h with $1 \leq h \leq l$.*

Then $cost(OPT) \leq l$.

Proof. From the description of the adversary we know that $|C_j| = n_j s_j$ and that the C_j's ($j \leq r$) are disjoint sets. Then there exists a bijection $\phi : U \to U$ such that for every $e \in U$, $\phi(e) \in C_j$ if and only if $e \in G_j$; this is because all sets C_j and all sets G_j are disjoint, $|C_j| = |G_j|$, for all j, and $\bigcup_{j=1}^{r} C_j = \bigcup_{j=1}^{r} G_j = U$. With a slight abuse of notation, denote by $\phi(P_h)$ ($h \leq l$) the set $\{\phi(e) \mid e \in P_h\}$. Note that $|\phi(P_h)| = d$. In addition, note that for every $i \leq r-1$, $|\phi(P_h) \cap \bigcup_{j=1}^{i} C_j| \geq d - s_{i+1}$. Hence, $\phi(P_h)$ defines a set which is not removed by the adversary in any phase, and since the collection of sets $\{\phi(P_1), \ldots, \phi(P_l)\}$ covers U, the cost of OPT is at most l. $\square$

We can now show that the cost of the optimal algorithm is upper-bounded by l, by showing that the conditions of Lemma 5 are satisfied.

Lemma 6 $cost(OPT) \leq l$.

Proof. Consider an instance of set cover, with a universe U of size n and the two subcollections of sets $\mathcal{S}_1$ and $\mathcal{S}_2$, as defined earlier in the section. Let G_i be the set of elements contained in the n_i sets of size s_i in $\mathcal{S}_1$, for $i \leq r$. Clearly, all G_i's are disjoint.

We will assume that all sets in $\mathcal{S}_2$ are of size d. This is not a restrictive assumption, as will become evident later. Let $\mathcal{S}_2$ consist of sets $P_1, \ldots, P_l$. We claim that $|P_h \cap \bigcup_{j=1}^{i} G_j| \geq d - s_{i+1}$, for all $i \leq r-1$ and $h \leq l$. To this end, we look at the choices of the greedy algorithm on the set cover instance that consists of sets in $\mathcal{S}_1 \cup \mathcal{S}_2$ over U. As argued by Slavík in [22] the greedy algorithm selects only sets in $\mathcal{S}_1$ in non-increasing order of size, breaking ties arbitrarily. Therefore, the greedy algorithm covers all elements in G_i prior to covering any element of $G_{i'}$, for all $i < i' \leq r$. Let the i-th phase of the greedy algorithm correspond to the selection of the n_i sets (each of size s_i) that cover the elements in G_i. Assume, by way of contradiction, that for one of the sets in $\mathcal{S}_2$, say P_h, $|P_h \cap \bigcup_{j=1}^{i} G_j| < d - s_{i+1}$. This implies that at the beginning of phase $i+1$ of the greedy algorithm, P_h covers at least $d - (d - s_{i+1}) + 1 = s_{i+1} + 1$ new elements. Since every set in $\mathcal{S}_1$ not yet selected by the greedy algorithm at the beginning of phase $i+1$ covers exactly s_{i+1} new elements, the greedy algorithm must select a set from $\mathcal{S}_2$, a contradiction.

We now show how to waive the assumption that all sets in $\mathcal{S}_2$ have size d. To every set $S \in \mathcal{S}_2$, of size $|S|$, we append $d - |S|$ elements from any set of size d that the greedy algorithm selects in its first phase. It is easy to see that the argument described earlier goes through. $\Box$

The following lemma, although not needed for the purposes of the proof of Theorem 8, will be essential to the proof of Theorem 9.

Lemma 7 (Appendix A) *For sufficiently large n, $cost(OPT) = l$.*

From Lemma 4 and Lemma 6, the approximation ratio of every priority algorithm for set cover is at least $\frac{\sum_{i=1}^{r} n_i}{l} = \frac{k}{l}$. From Slavík's analysis, this is precisely the approximation ratio of the greedy set cover algorithm which shows:

Theorem 8 *No priority algorithm for set cover performs better than* the *greedy algorithm. The approximation ratio for priority set cover is thus* $\ln n - \ln\ln n + \Theta(1)$ *and this bound is tight.*

4.3 Adaptive priority metric facility location: memoryless algorithms

The result of Theorem 8 has important implications for the approximability of metric facility location by priority algorithms. First, note that Guha and Khuller [8] established a polynomial time reduction from set cover to metric facility location, shown in Figure 1 (we have modified their notation for consistency with our paper.). For the set cover problem, U is the universe and $\mathcal{S}$ is the collection of sets. The corresponding facility location problem $(\mathcal{F}, \mathcal{C})$ has the set of cities $\mathcal{C} = U$ and for each set $S \in \mathcal{S}$ there is a corresponding facility $F \in \mathcal{F}$ where the distance c_{ij} from facility i to city j is set to 1 (respectively 3) if $x_j \in S_i$ (resp. $x_j \notin S_i$).

Create a facility location instance $(\mathcal{F}, \mathcal{C})$ corresponding to $(\mathcal{S}, U)$
Let $f = \gamma\frac{|\mathcal{C}|}{l}$ % the cost of a facility
while $\mathcal{C} \neq \emptyset$ **do**
    $\mathcal{F}'$ = Adaptive-Priority-Algorithm $(\mathcal{F}, \mathcal{C})$
    Let $\mathcal{C}'$ be the cities covered at distance 1
    Let $\mathcal{F} = \mathcal{F} - \mathcal{F}'$ and $\mathcal{C} = \mathcal{C} - \mathcal{C}'$
    Let $f = \gamma\frac{|\mathcal{C}|}{l}$ be the cost of a facility
endwhile

Figure 1: The Guha and Khuller reduction. The priority facility location algorithm returns a set of opened facilities. Here γ is an appropriately chosen constant and ℓ is the optimal cost for the set cover instance.

Using the reduction, Guha and Khuller argued that if there exists a polynomial-time algorithm for metric facility location with approximation ratio better than 1.463, then set cover can be approximated (by some polynomial-time algorithm) with approximation ratio $(1-\epsilon)\ln n$, for $\epsilon > 0$, where n is the size of the universe. By Feige's inapproximability result for set cover [7], this implies that $NP \subseteq DTIME(n^{O(\log\log n)})$. In fact, the Guha-Khuller reduction can be used to show that one can derive a priority algorithm for set cover from any *memoryless* priority algorithm. In view of this observation, we can interpret the argument of Guha and Khuller as follows: If there exists a memoryless priority algorithm for metric facility location with approximation ratio better than 1.463, then set cover is $(1-\epsilon)\ln n$-approximable by a priority algorithm for some $\epsilon > 0$, contradicting Theorem 8.

As can be seen in Figure 1, the set cover algorithm consists of several "stages"; in each stage the facility location algorithm is executed for an opening cost[7] which is uniform for this stage. We can assume, from Lemma 7 that for sufficiently large $n = |U|$, the optimal cost l is known by the set cover priority algorithm. For a given stage of the reduction, the facility location algorithm returns a set of facilities to be opened. Given this set of open facilities, the stage ends by removing the corresponding sets and also removing the corresponding elements of the set cover universe for each city covered (i.e. at distance 1) by a facility opened during this stage. The stage ends by uniformly resetting the opening cost for the facilities.

We first provide more details as to how one stage in this reduction is applied in the context of priority algorithms. We note that similar to the discussion in [5], any adaptive memoryless facility location algorithm A can be transformed to a memoryless greedy algorithm. For our purposes we want a slightly different transformation, namely we want to transform A to an algorithm A' that opens the same set of facilities, and has the following property: if A' considers a facility that it does not open, none of the remaining (i.e. not yet considered) facilities will be opened. More precisely, A' simulates A, in the sense that every time A would open a facility, A' will also consider and open the facility in question. If, however A' would not open a certain facility, then A' will give this facility the *lowest* priority thereby deferring its consideration. We also need to make sure that such a facility, if it was considered (i.e. had highest priority in some iteration) and not opened by A, will continue to receive lowest priority and never be opened in subsequent iterations of A'. This is easy to do, since A is a memoryless algorithm: as A' starts a new iteration, we can reconstruct the order in which A has opened facilities before this iteration, say in the order $F_1, \ldots, F_r$. We consider each of the configurations $\emptyset, \{F_1\}, \{F_1, F_2\}, \ldots, \{F_1, \ldots F_{r-1}\}$. Then in determining the priority of any facility F, we consider its priority given each of these configurations. For example, in configuration $\{F_1, \ldots F_{i-1}\}$, if A would have given F higher priority than the facility F_i that was next opened but would not have opened F, then A' will give lowest priority to facility F.

Assume, therefore, that the facility location algorithm has the property stated above. We can now make each iteration of the facility location algorithm correspond to an iteration of a set cover algorithm, and in particular describe how to order sets in each iteration. For definiteness, think of the priority of a facility being given by a non-negative real number, with priority zero indicating that the algorithm should not open this facility. Now we indicate how to give priorities to sets so that each iteration of the facility location algorithm will correspond to a priority set cover iteration. The priority of a set S_i (in each iteration of the facility location algorithm) will be given by a pair (a_i, b_i) where a_i is the priority that the facility location algorithm would give to the i^{th} facility (that is, the facility to which S_i corresponds) if we were not beginning a new stage and b_i is the priority that would be given to the i^{th} facility if we were beginning a new stage. In the latter case these priorities depend upon the new opening costs as well as knowing what cities and facilities have been removed. These pairs are then ordered lexicographically so that $(0, b)$ indicates a lower priority than (a, b') for any $a > 0$ and $(0, b)$ indicates lower priority than $(0, b')$ if $b < b'$. Hence, using this reduction, the set covering algorithm will not "consider" a set it would not open.

We thus established the following:

Theorem 9 *Metric facility location cannot be approximated within a factor smaller than 1.463 by any memoryless priority algorithm.*

[7] To derive the 1.463 hardness result for facility location, the constant γ is set to .463.

5 Fixed priority algorithms

In this section we first present a tight lower bound on the approximation ratio of algorithms in the class Fixed Priority, Greedy for metric facility location. The construction behind the proof also suggests strong lower bounds for Fixed Priority set cover, and Fixed Priority Greedy facility location in arbitrary spaces, as well as a lower bound that is better than 4/3 for Fixed Priority (but not necessarily Greedy) metric facility location algorithms with unbounded memory.

Consider the following instance $(\mathcal{F}, \mathcal{C})$ of the metric facility location problem. Let $C_1, C_2, \ldots, C_d$ be a partition of the n cities into d (disjoint) sets of size $\frac{n}{d}$ each, for some large constant d. For every pair (k,l), with $1 \leq k, l \leq d$, and $k \neq l$ we identify the sets of facilities $F_{k,l}$ and $\tilde{F}_{k,l}$ as follows. Denote by $f_{ij}^{k,l}$ the facility that covers every city in C_k, except for city $i \in C_k$, and also covers city $j \in C_l$. In addition, denote by $\tilde{f}_{ij}^{k,l}$ the facility that covers every city in C_l with the exception of city $j \in C_l$, and also covers city $i \in C_k$. $F_{k,l}$ and $\tilde{F}_{k,l}$ are defined as the sets $\{f_{ij}^{k,l} \mid i \in C_k, j \in C_l\}$ and $\{\tilde{f}_{ij}^{k,l} \mid i \in C_k, j \in C_l\}$, respectively. Note that by definition, the complement wrt $C_k \cup C_l$ of a facility in $F_{k,l}$ is in $\tilde{F}_{k,l}$ (and vice versa). The cost for connecting every city in $\mathcal{C}$ not covered by a facility in $F_{k,l}$ (respectively, $\tilde{F}_{k,l}$) to a facility in $F_{k,l}$ (respectively, $\tilde{F}_{k,l}$) is set to 3. The set of potential facilities $\mathcal{F}$ is defined as the union of all $F_{k,l}$ and $\tilde{F}_{k,l}$, for all pairs k,l, with $k \neq l$. Every facility is assigned an opening cost of $2 - \epsilon$. We emphasize that all connection costs are in $\{1, 3\}$ and that the facilities have uniform opening costs.

Let σ be the ordered sequence of facilities in $\mathcal{F}$ produced by a Fixed Priority Greedy algorithm.

Lemma 10 *For every pair k,l, with $1 \leq k,l \leq d$ and $k \neq l$, there exists a set $S_{k,l}$ of $\frac{n}{d}$ pairs of facilities $(f_1, \overline{f}_1), \ldots, (f_{\frac{n}{d}}, \overline{f}_{\frac{n}{d}}) \in (F_{k,l} \times \tilde{F}_{k,l}) \cup (\tilde{F}_{k,l} \times F_{k,l})$ such that: For every m with $1 \leq m \leq \frac{n}{d}$, the facilities f_m and $\overline{f}_m$ are complementary wrt $C_k \cup C_l$, f_m precedes $\overline{f}_m$ in σ, and at least one of the following holds:*

1. *Each f_m covers a city in C_k which is not covered by any facility of the form $f_{m'} \in S_{k,l}$ with $m' \neq m$; or*
2. *Each f_m covers a city in C_l which is not covered by any facility of the form $f_{m'} \in S_{k,l}$ with $m' \neq m$.*

Proof. Fix k,l. Suppose that for every city $j \in C_l$ there exists city $i_j \in C_k$ such that $f_{i_j j}^{k,l}$ precedes $\tilde{f}_{i_j j}^{k,l}$ in σ. Let $S_{k,l}$ be the set of $\frac{n}{d}$ pairs of facilities $(f_{i_j j}^{k,l}, \tilde{f}_{i_j j}^{k,l})$ for $j \in C_l$, then the lemma holds, since $f_{i_j j}^{k,l}$ and $\tilde{f}_{i_j j}^{k,l}$ are complementary wrt $C_k \cup C_l$, and $f_{i_j j}^{k,l}$ covers only j among the cities in C_l. Otherwise, there exists city $j' \in C_l$ such that for every city $i \in C_k$ $\tilde{f}_{ij'}^{k,l}$ precedes $f_{ij'}^{k,l}$ in σ. Let $S_{k,l}$ be the set of $\frac{n}{d}$ pairs of facilities $(\tilde{f}_{ij'}^{k,l}, f_{ij'}^{k,l})$, then $\tilde{f}_{ij'}^{k,l}$ and $f_{ij'}^{k,l}$ are complementary wrt $C_k \cup C_l$, and $\tilde{f}_{ij'}^{k,l}$ covers only i among the cities in C_k. $\square$

Note that for every fixed pair k,l, either case (1) or case (2) of Lemma 10 (or possibly both) apply. For the purposes of our proof we will assume, without loss of generality, that only one of the cases applies, for every fixed pair k,l (if both cases apply, consider only one arbitrarily). We use the notation $C_l \to C_k$ (resp. $C_k \to C_l$) to denote that case (1) (resp. case (2)) applies.

Define a digraph $G = (V, E)$ on d vertices $v_1, \ldots, v_d$ as follows: the directed edge (v_k, v_l), with $k \neq l$ is in E iff $C_k \to C_l$. From Lemma 10, any two vertices in G are adjacent to one common directed edge, and thus G has exactly $\binom{d}{2}$ edges. The following lemma is straightforward.

Lemma 11 *Let G be defined as above. There exists a set $V' \subseteq V$ of size at most $\log d$ that* dominates $V \setminus V'$. *Namely, for every $v_l \in V \setminus V'$, there exists $v_k \in V'$ such that $(v_k, v_l) \in E$.*

Lemma 11 implies that there exists a collection D of at most $\log d$ sets $C_{k_1}, \ldots, C_{k_p}$ such that for every set $C_l \notin D$, there exists $C_{k_i} \in D$ such that $C_{k_i} \to C_l$. We now describe how the adversary constructs the input that is presented to the algorithm. Recall from Lemma 10 that the set $S_{k_i,l}$ of facilities satisfies $C_{k_i} \to C_l$. For fixed l, with $C_l \notin D$, denote by I_l the set $S_{k_i,l}$ with the property that i is the minimum among all j's for which $C_{k_j} \to C_l$, and $C_{k_j} \in D$. Let $I = \bigcup\{I_l \,|C_l \notin D\}$. The input to the algorithm contains only facilities in I. In addition, for every $f \in I$ which belongs in $S_{k_i,l}$, the adversary removes f's complement wrt $C_{k_i} \cup C_l$, except for the case when f and its complement are the last pair of complementary facilities wrt $C_{k_i} \cup C_l$. Note that from Lemma 10, f's complement wrt $C_{k_i} \cup C_l$ follows f in σ, so the removal of the facilities by the adversary is feasible. All facilities $I' \subset I$ which are not explicitly removed by the adversary comprise the actual input.

It remains to bound the cost of the priority algorithm and the optimal algorithm. From the construction of I' and Lemma 10, it follows that the first facility that can cover a city $j \in \mathcal{C} \setminus D$ covers only j among the cities in $\mathcal{C} \setminus D$. The greedy criterion dictates that when the algorithm considers the facility in question, it will open it: the algorithm will pay a total of $3 - \epsilon$ for opening the facility and connecting j to the open facility, which improves upon the cost of 3 that must be paid if the algorithm does not open the facility. Since there are $\frac{n}{d}(d - |D|) \geq \frac{n}{d}(d - \log d)$ cities in $\mathcal{C} \setminus D$ (where $|D|$ is the size of D), we get

$$cost(ALG) = (2 - \epsilon) \cdot \frac{n}{d} \cdot (d - |D|) + n \geq (2 - \epsilon) \cdot \frac{n}{d} \cdot (d - \log d) + n.$$

The optimal algorithm, on the other hand, opens only pairs of facilities that are complementary with respect to partitions of cities. The open facilities cover all cities in the instance, and hence the total optimal cost is

$$cost(OPT) \leq 2 \cdot (2 - \epsilon) \cdot (d - |D|)d + n \leq 2 \cdot (2 - \epsilon) \cdot d^2 + n.$$

Observe that the ratio $\frac{cost(ALG)}{cost(OPT)}$ can be made arbitrarily close to 3, for large, albeit constant d. We thus showed the following:

Theorem 12 *The approximation ratio of every* Fixed Priority, Greedy *algorithm for metric facility location is at least $3 - \epsilon$, for arbitrarily small ϵ.*

Using an argument along the same lines, one can derive the following.

Theorem 13 *The approximation ratio of every* Fixed Priority, Greedy *algorithm for facility location in arbitrary spaces (resp.* Fixed Priority *algorithm for set cover) is at least $(1 - \epsilon)n$, where n is the number of cities (resp. the size of the universe) in the input instance.*

Proof sketch: For facility location in arbitrary spaces, apply the construction behind the proof of Theorem 12, with every distance of 3 replaced by the infinite distance. We choose a uniform facility cost that is sufficiently large, that is, much larger than n, but not infinite. Similarly, for set cover, replace "facilities" by "sets" and "cities" by "elements"; an element is covered by a set if the corresponding city is at distance 1 from the corresponding facility in the facility-location instance. The proof of Theorem 12 shows that the algorithm will open $(1 - \epsilon)n$ facilities (resp. sets). □

What can be said about Fixed Priority algorithms for metric facility location that are not necessarily greedy? While we are not aware of tight bounds for this class of algorithms, we show how the construction behind the proof of Theorem 12 suggests a lower bound that is (slightly) better than the 4/3 bound shown in Theorem 1 (and which applies to priority algorithms with unbounded memory).

Recall the construction of the set of facilities I, as shown earlier in this section. Let g be the uniform facility cost (that is, we replace the $2-\epsilon$ facility cost by g, to be determined later). As in the case of Greedy algorithms, the input to the Fixed Priority algorithm contains only facilities in I, and for every $f \in I$ which belongs in I_l, the adversary removes f's complement wrt $C_{k_i} \cup C_l$ (for some appropriate k_i) except for the case f and its complement are the last pair of complementary facilities wrt $C_{k_i} \cup C_l$. In the latter case, we call f *critical wrt* C_l, because the adversary's decision on whether to remove f's complement wrt $C_{k_i} \cup C_l$ is made according to the following rule. Let x_l be the fraction of the $|C_l| = \frac{n}{d}$ facilities in $S_{k_i,l}$ that the algorithm opened right after f is considered. Then, the algorithm will not remove f's complement wrt $C_{k_i} \cup C_l$ *if and only if*

$$\frac{g \cdot x_l|C_l| + x_l|C_l| + 3(1-x_l)|C_l|}{g|C_l| + |C_l|} \leq \frac{g \cdot x_l|C_l| + |C_l|}{2g + |C_l|}. \tag{1}$$

Let us give some intuition about the actions of the adversary. The numerator of the LHS of (1) is the cost that the algorithm will pay to accommodate cities in C_l, in the case when the complement wrt $C_{k_i} \cup C_l$ of the critical facility wrt C_l is removed; in such a case, the corresponding optimal cost is given by the denumerator. On the other hand, the cost of the algorithm if the said facility is not removed, is precisely the numerator of the RHS of (1) (and, likewise, the denumerator is the corresponding optimal cost). Thus, if (1) holds, it makes sense for the adversary to not remove the complement of the critical facility. Since a large fraction of cities (arbitrarily close to 1) belong to some $C_l \notin D$ (where D is as defined earlier in the section), the total cost will be "dominated" by the cost paid to accommodate such cities.

To lower-bound the cost of the algorithm, partition $C \setminus D$ into two disjoint collections of sets of cities, denoted by C_{in}, C_{out}. C_{in} contains all $C_l \notin D$ for which the last critical complement (wrt $C_{k_i} \cup C_l$ for some appropriate i) in I_l was not removed, while C_{out} contains all $C_l \notin D$ for which the said complement was in fact removed by the adversary. Then, the cost of the algorithm can be lower-bounded as follows:

$$cost(ALG) \quad \geq \quad \sum_{l:C_l \in C_{out}} (g \cdot x_l|C_l| + x_l|C_l| + 3(1-x_l)|C_l|) + \sum_{l:C_l \in C_{in}} (g \cdot x_l|C_l| + |C_l|). \tag{2}$$

On the other hand, optimal cost is as follows:

$$cost(OPT) \quad \leq \quad \sum_{l:C_l \in C_{out}} (g|C_l| + |C_l|) + \sum_{l:C_l \in C_{in}} (2g + |C_l|) + 3\frac{n}{d}\log d, \tag{3}$$

where the term $3\frac{n}{d}\log d$ upper-bounds the connection cost of cities in D. To facilitate the exposition, we can ignore this term, since for any arbitrarily small constant $\epsilon > 0$, we can find sufficiently large d such that the term is smaller than $\epsilon \cdot n$. (recall that the optimal cost, is, by comparison, considerably larger than this term, namely at least n). The definition of our adversary, along with inequalities (2) and (3) imply that the approximation ratio is minimized if, for every l, x_l is such as

$$\frac{g \cdot x_l|C_l| + x_l|C_l| + 3(1-x_l)|C_l|}{g|C_l| + |C_l|} = \frac{g \cdot x_l|C_l| + |C_l|}{2g + |C_l|}. \tag{4}$$

Since at the end we will select g to be a small constant, for large n we will have that $g << |C_l|$, which means that we can ignore the term $2g$ in the denumenator of the RHS of (4), thus (4) gives $x_l = \frac{2-g}{g^2+2}$. Substituting in (4), we have that the approximation ratio is at least $\frac{2+2g}{2+g^2}$, which is maximized for $g = 0.7321$, giving an approximation ratio of 1.366.

Theorem 14 *The approximation ratio of every* FIXED PRIORITY *algorithm for metric facility location is at least* $1.366 - \epsilon$, *for arbitrarily small* ϵ.

6 Future directions and open problems

Several interesting issues and open problems are left to investigate. A natural direction is to improve the lower bound for adaptive-priority metric facility location, if this is indeed possible. Can the memoryless assumption be removed in the 1.463 lower bound? Similarly, although the $3 - \epsilon$ bound we showed is tight for the class FIXED PRIORITY GREEDY, we do not know whether a bound better than 1.366 can be obtained for FIXED PRIORITY (not necessarily greedy) metric facility location algorithms. Our lower-bound constructions use a $\{1, 3\}$ cost metric, and all instances are unweighted. Can we improve the results by considering arbitrary metric distances or weighted instances, where each city has a weight and the connection cost from a city is the product of the distance and the weight?

As already discussed, there is another natural way to model facility location priority algorithms, namely by letting the cities be the input items. Meyerson [19] gives a randomized $O(1)$-approximation priority algorithm in this model where both the ordering and the irrevocable decisions use randomization. Does there exist a deterministic $O(1)$-approximation priority algorithm in this setting? More generally, we need to study the power of randomization in the context of priority algorithms (see [1] for a more elaborate discussion).

There are a number of variants of the facility location problem as well as the related k-median problem that can be studied within our framework. For example, the capacitated facility location problem where each facility has a capacity bound on the number (weight) of the cities it can serve. Of course, our lower bounds apply to the capacitated version; that is, by setting all capacities to exceed the sum of all city weights. But can we derive better results for this variant? As in the set cover problem, it seems appropriate to only consider greedy priority algorithms since not opening a facility may result in an infeasible solution. We are also considering the k-facility location problem in which a feasible solution allows at most k opened facilities, generalizing the facility location and k-median problems.

Acknowledgments

This work began in discussions with Yuval Rabani and our first result, namely Theorem 1, was a result of that collaboration. We also thank Éva Tardos and David Williamson for their suggestions and references, as well as Joan Boyar and Kim Larsen for their very helpful comments on Theorem 9.

References

[1] S. Angelopoulos. Randomized priority algorithms. Manuscript, 2003.

[2] S. Arora, B. Bollobás, and L. Lovász. Proving integrality gaps without knowing the linear program. In *Proceedings of the 43rd Annual IEEE Conference on Foundations of Computer Science*, pages 313–322, 2002.

[3] S. Arora and M. Sudan. Improved low degree testing and its applications. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 485–495, 1997.

[4] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating throughput in real-time scheduling. *SIAM Journal of Computing*, 31(2):331–352, 2001.

[5] A. Borodin, M. Nielsen, and C. Rackoff. (Incremental) priority algorithms. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 752–761, 2002.

[6] V. Chvátal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[7] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[8] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.

[9] S. Guha, N. Mishra, R. Motwahl, and L. O'Callaghan. Clustering data streams. In *Proceedings of the 41th Annual IEEE Conference on Foundations of Computer Science*, pages 359–366, 2000.

[10] D. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.

[11] R. Impagliazzo and S. Davis. models of greedy algorithms for graph problems. Manuscript, 2002.

[12] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computation*, pages 731–740, 2002.

[13] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

[14] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1999.

[15] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.

[16] M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. A greedy facility location algorithm analyzed using dual fitting. In *Proceedings of the 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 127–137, 2001.

[17] M. Mahdian, J. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 229–242, 2002.

[18] R. R. Mettu and C. G. Plaxton. The online median problem. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, 2000.

[19] A. Meyerson. Online facility location. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.

[20] D.B. Shmoys. Approximation algorithms for facility location problems. In K. Jansen and S. Khuller, editors, *Approximation Algorithms for Combinatorial Optimization*, volume 1913 of *Lecture Notes in Computer Science*. Springer, Berlin, 2000.

[21] D.B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[22] P. Slavík. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25:237–254, 1997.

Appendix

A Proofs of Some Lemmas

Proof of Lemma 7.

Given Lemma 6, suffices to show that $cost(OPT) \geq l$, since it follows that $cost(OPT) = l$. We can assume that $n = N(k,l)$, since the lower-bound construction in [22] holds for all $n \geq N(k,l)$. In this case, the size of the largest set in $\mathcal{S}_1 \cup \mathcal{S}_2$ (denoted by q_1 in [22]) is *exactly* $\lceil \frac{n}{l} \rceil$ (this follows from (4), (5) and (11) in [22]). Suppose, by way of contradiction, that $cost(OPT) < l$. Since $\lceil \frac{n}{l} \rceil$ is an upper bound on the largest size of a set in the instance (as constructed by the adversary), this would imply $\lceil \frac{n}{\lceil \frac{n}{l} \rceil} \rceil \leq l-1$, which implies that $\frac{nl}{n+l} \leq l-1$, or equivalently $n \leq l^2 - l$. However, n grows exponentially with l (see Eq.(36) in [22]), which leads to a contradiction for sufficiently large n. $\square$

Proof of Lemma 11. We show how to construct V'. Initially $V' = \emptyset$. Let v_1 be a vertex of largest outdegree in G; clearly, the outdegree of v_1 is at least $\lceil \frac{\binom{d}{2}}{d} \rceil = \lceil \frac{d-1}{2} \rceil$. Place v_1 in V' and repeat the process in the digraph induced by $V \setminus \{N^+(v_1) \cup v_1\}$, where $N^+(v_1)$ is the out-neighborhood of v_1. Note that any two vertices of the induced digraph are adjacent to one common directed edge, and that the induced graph has at most $d-1-\lceil \frac{d-1}{2} \rceil \leq \frac{d-1}{2}$ vertices. It is easy to see that by repeating the process at most $\log d$ times, one can find a set V' of size at most $\log d$ that dominates $V \setminus V'$. $\square$