

## The Power of Priority Algorithms for Facility Location and Set Cover

Spyros Angelopoulos<sup>1</sup> and Allan Borodin<sup>1</sup>

**Abstract.** We apply and extend the priority algorithm framework introduced by Borodin, Nielsen, and Rackoff to define “greedy-like” algorithms for the (uncapacitated) facility location problems and set cover problems. 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 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.

**Key Words.** Greedy algorithms, Approximation lower bounds.

**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 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 et al. [6] 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 [6] 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

---

<sup>1</sup> Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4. {spyros,bor}@cs.toronto.edu.

(weighted) set cover problem is achieved by a most natural greedy algorithm [15], [17], [7]. Likewise, the best known approximation ratio for the metric uncapacitated facility location problem, namely 1.52, which is due to Mahdian et al. [19] is achieved by a greedy-like algorithm.<sup>2</sup> In addition, quite good approximation ratios for facility location (namely, 1.61 [13] and 1.861 [18]) have also been derived by greedy-like algorithms. 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.

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 [6], 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 it has already considered has been opened).

As in [6], “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

---

<sup>2</sup> Even though the algorithm of [19] works in two phases, and at first sight does not appear to follow the priority framework, it can be formulated as a priority algorithm.

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 [24] 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 [9] 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  $(1 - \varepsilon) \ln n$  for any  $\varepsilon > 0$ .
  - (b) For any  $\varepsilon > 0$ , no FIXED PRIORITY algorithm can achieve an approximation ratio better than  $(1 - \varepsilon)n$ .

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  $\frac{4}{3}$ .
  - (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 [10] (under the same assumption used by Feige [9]).
  - (c) For all  $\varepsilon > 0$ , no FIXED PRIORITY GREEDY algorithm can achieve an approximation ratio of  $3 - \varepsilon$  for the uniform metric facility location problem. This bound matches the 3-approximation upper bound of Mettu and Plaxton [20] 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 [15], [17], [7] as well as the greedy-like algorithms for facility location due to Mahdian et al [18], [19] and Jain et al. [13] can all 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, a reverse-reconstruction step) do not appear to belong to the class of priority algorithms. Hence, a negative result concerning a priority algorithm provides some evidence that a reverse-reconstruction phase may be 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 [6], 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 et al. [5] 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, a 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  $\mathcal{F} \cap \mathcal{C} = \emptyset$ .<sup>3</sup> 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 a 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 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 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.

<sup>3</sup> 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  $x_c$ .

For the class of scheduling problems considered in [6], there is no issue as to what the “input items” are. However, 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 [21], 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. Indeed this model is very natural in the sense that one would expect the 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 [15], [17] observes the above requirement. It is worth mentioning that in this context every priority set cover algorithm belongs to the class of GREEDY algorithms.<sup>4</sup> 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 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 online 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 choose an optimal ordering for each input (e.g., choose 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.

---

<sup>4</sup> 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.

We consider the basic framework where the priority algorithm has no additional global information (such as the total number of input items,<sup>5</sup> 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 a concrete example, suppose that a priority algorithm specifies the ordering  $\sigma = i_1, i_2, \dots, i_n$  for  $n$  input items  $I = \{i_1, \dots, i_n\}$ , with  $i_1$  having the highest and  $i_n$  the lowest priority, respectively. Then, for every subset  $I' \subseteq I$ , the same priority algorithm will order  $I'$  in such a way that input item  $i_{j_k}$  precedes input item  $i_{j_l}$  (assuming  $i_{j_l}, i_{j_k} \in I'$ ) if  $i_{j_k}$  precedes  $i_{j_l}$  in  $\sigma$ . We do not allow an algorithm the power to produce orderings that do not comply with the above rule.

To enforce the above constraint on the allowable orderings, 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 an input item (facility), the priority of every other input item is unaffected.<sup>6</sup>

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) to derive the actual input  $S'$ .

Following the discussion in [6], 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 et al. [23] 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 [22] 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 et al. [19], and is achieved by a memoryless algorithm that can be captured by the priority framework. For the special case of connection costs in  $\{1, 3\}$ , the LP-based (non-priority) algorithm of Guha and

<sup>5</sup> 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.

<sup>6</sup> This explains why we restrict our attention to inputs in which the set of facilities and the set of cities are disjoint, as mentioned earlier.

Khuller [10] provides a 1.463-approximation, matching the complexity-based hardness result presented in the same paper.

We identify other algorithms that follow the paradigm of a priority algorithm, as defined in Section 2.2. Hochbaum [12] 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 this particular algorithm is tight. Mahdian et al. [18] showed that a natural (memoryless) 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. Another memoryless ADAPTIVE PRIORITY algorithm is due to Jain et al. [13]: their algorithm belongs in the class ADAPTIVE PRIORITY as well, and it is also analyzed by using the dual-fitting technique. On the other hand, Mettu and Plaxton [20] showed that an algorithm which belongs in the class FIXED PRIORITY GREEDY yields a 3-approximation for the problem. It should be mentioned that their algorithm is in fact a re-statement of a primal-dual algorithm due to Jain and Vazirani [14].

Set cover is one of the oldest and most well-studied  $NP$ -hard problems. Johnson [15] and Lovász [17] proposed a simple greedy algorithm which they showed provides an  $H(n)$ -approximation, where  $H(n)$  is the  $n$ th harmonic number, and  $n = |U|$ . Chvátal [7] extended their results to the weighted case. A tight analysis of the greedy algorithm due to Slavík [24] is of particular interest, as discussed in Section 3.2. 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 [9] showed that it is not possible to obtain a polynomial-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 [4] 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 et al. [6] does not seem to be directly applicable to graph optimization problems without certain modifications. In recent work, Davis and Impagliazzo [8] addressed this issue and proposed an extension of the framework 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 an integrality gap better than  $2 - \epsilon$ , Arora et al. [3] recently showed that this bound is tight for three broad classes of IP formulations for this problem. The results suggest that every linear relaxation 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 et al. [16] 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., [11]). 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.

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  $\bar{f}$  are *complementary*, where  $\bar{f}$  is the complement of  $f$  with respect to  $\mathcal{C}$ .

**3.1. 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 3.3 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  $|C| = n/2$ , there exists a facility  $f$  that covers  $C$  (and only cities in  $C$ ).<sup>7</sup> 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  $\bar{f}$  of  $f$  is also in the instance.

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  $\bar{f}$  unless  $\bar{f}$  is the last remaining facility in the input set. If the algorithm does not open  $f$ , the adversary removes all remaining facilities except for  $\bar{f}$ . Let all facility costs be equal to  $n/4$ . The optimal algorithm will open a pair of complementary facilities at an opening cost of  $2 \cdot (n/4)$ , 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  $k \cdot (n/4)$  and a connection cost of at least  $n(1 - 2^{-k}) + 3n \cdot 2^{-k}$ , and thus a total cost of  $(1 + k/4 + 2^{-k+1})n$ . 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.  $\square$

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  $(\log n)/2$ .*

<sup>7</sup> Throughout this paper we ignore floors and ceilings and assume that  $n$  is appropriately divisible.

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  $\infty$  distance from  $f$ . In this case any priority algorithm must open every facility it considers. Hence it will consider and open  $\log n$  facilities while the optimal algorithm will again open two facilities.  $\square$

Note that the bound is tight, since Hochbaum [12] showed that a simple greedy-like algorithm, which can be classified as ADAPTIVE PRIORITY, is an  $O(\log n)$  approximation algorithm for the problem.

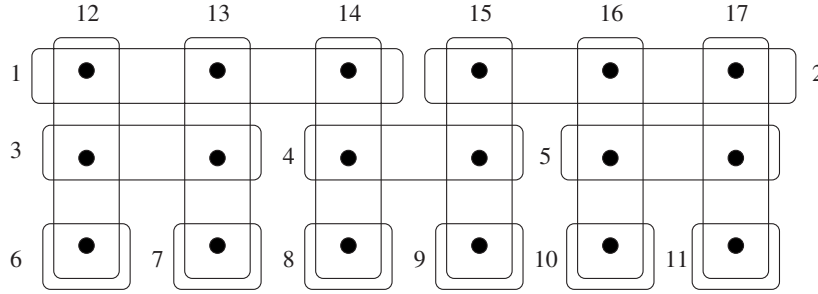
**3.2. Adaptive Priority Set Cover.** In this section we prove that no (adaptive) priority algorithm can perform better than *the* greedy set-cover algorithm.

First, note that a simple argument along the lines of Theorem 2 shows a lower bound of  $(\log n)/2$  on the approximation ratio of every priority algorithm for set cover. Even though this bound is *asymptotically* optimal, in this section we will show a tighter bound, namely, a bound of  $\ln n - o(\ln n)$ . The improved bound has important implications on the approximability of metric facility location, as will become clear in Section 3.3.

A remarkably tight analysis of the greedy set cover algorithm due to Slavík [24] 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  $n_1, n_2, \dots, n_r$  disjoint sets of sizes  $s_1, s_2, \dots, s_r$ , 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 [24] appropriately, but for the purpose 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 positive integer; (ii)  $\sum_{i=1}^r n_i = k$ ; (iii)  $s_1 > s_2 > \dots > s_r > 0$ ; and (iv)  $s_1 \geq \lceil n/l \rceil$ . The second subcollection, denoted by  $\mathcal{S}_2$ , consists of  $l$  sets of sizes in  $\{\lceil n/l \rceil, \lceil n/l \rceil - 1\}$ . 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  $k/l \geq \ln n - \ln \ln n + \Theta(1)$ . Figure 1 shows an example of Slavík's construction.

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. In other words, we will present an adversary which forces every priority algorithm to "simulate" the actions of *the* greedy algorithm on an instance that resembles Slavík's original one for the greedy 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  $d = s_1 \geq \lceil n/l \rceil$ . All sets have identical costs. Given an adaptive priority algorithm for the problem, we describe the actions of the adversary,



**Fig. 1.** Slavik's construction for a universe of size  $n = 18$ . Here  $N(k, l) = n = 18$ ,  $(n_1, n_2, n_3) = (2, 3, 6)$ , and  $(s_1, s_2, s_3) = (3, 2, 1)$ . The greedy algorithm will consider and select all sets in  $\mathcal{S}_1 = \{1, \dots, 11\}$  (thus  $k = 11$ ), while the optimal algorithm selects all sets in  $\mathcal{S}_2 = \{12, \dots, 17\}$  (thus  $l = 6$ ).

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 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  $P_1, P_2, \dots, P_{n_i}$  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  $S$  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  $R_j$ . To this end, note that for all  $j < i$ ,

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

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  $\text{cost}(\text{ALG})$ ,  $\text{cost}(\text{OPT})$  the cost of the priority algorithm and the cost of the optimal algorithm, respectively. We then have:

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

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  $P_1, P_2, \dots, P_l \subseteq U$ , each of size  $d$ , with the following properties:*

- $P_1 \cdots P_l$  cover all elements of  $U$ , namely,  $\bigcup_{h=1}^l P_h = U$ .
- There exist disjoint sets  $G_1, G_2, \dots, G_r \subseteq U$ , 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  $\text{cost}(\text{OPT}) \leq l$ .

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  $\varphi : U \rightarrow U$  such that for every  $e \in U$ ,  $\varphi(e) \in C_j$  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  $\varphi(P_h)$  ( $h \leq l$ ) the set  $\{\varphi(e) \mid e \in P_h\}$ . Note that  $|\varphi(P_h)| = d$ . In addition, note that for every  $i \leq r - 1$ ,  $|\varphi(P_h) \cap \bigcup_{j=1}^i C_j| \geq d - s_{i+1}$ . Hence,  $\varphi(P_h)$  defines a set which is not removed by the adversary in any phase, and since the collection of sets  $\{\varphi(P_1), \dots, \varphi(P_l)\}$  covers  $U$ , the cost of  $\text{OPT}$  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.  $\text{cost}(OPT) \leq l$ .

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 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  $P_1, \dots, P_l$ . 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 [24] 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.  $\square$

The following lemma, although not needed for the purposes of the proof of Theorem 8, will be essential to the proof of Theorem 9. The lemma states that one can consider instances in which the optimal set cover cost is *exactly*  $l$ . We emphasize that the lower bound we show in this section holds even assuming that the algorithm has knowledge of the optimal cost. Hence we can assume that this cost, namely  $l$ , is known to the algorithm.

LEMMA 7. *For sufficiently large  $n$ ,  $\text{cost}(OPT) = l$ .*

PROOF. See the Appendix.

From Lemmas 4 and 6, the approximation ratio of every priority algorithm for set cover is at least  $(\sum_{i=1}^r n_i)/l = k/l$ . 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.*

```

Create a facility location instance  $(\mathcal{F}, \mathcal{C})$  corresponding to  $(\mathcal{S}, U)$ 
Let  $f = \gamma|\mathcal{C}|/\ell$  be the cost of a facility
while  $\mathcal{C} \neq \emptyset$  do
     $\mathcal{F}' = \text{Adaptive-Priority-Algorithm}(\mathcal{F}, \mathcal{C})$ 
    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  $f = \gamma|\mathcal{C}|/\ell$ 
endwhile

```

**Fig. 2.** The Guha and Khuller reduction. The priority facility location algorithm returns a set  $\mathcal{F}'$  of opened facilities. Here  $\gamma$  is an appropriately chosen constant (in particular,  $\gamma = 0.463$ ) and  $\ell$  is the optimal cost for the set cover instance.

**3.3. 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, as shown in Figure 2, note that Guha and Khuller [10] established a polynomial-time reduction from set cover to metric facility location (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 (resp. 3) if  $x_j \in S_i$  (resp.  $x_j \notin S_i$ ). We denote by  $\mathcal{F}'$  the set of facilities opened by the priority facility location algorithm.

Using the reduction, Guha and Khuller argued that if there exists a polynomial-time algorithm for metric facility location with an approximation ratio better than 1.463, then set cover can be approximated (by some polynomial-time algorithm) with an approximation ratio  $(1 - \varepsilon) \ln n$ , for  $\varepsilon > 0$ , where  $n$  is the size of the universe. By Feige’s inapproximability result for set cover [9], this implies that  $NP \subseteq DTIME(n^{O(\log \log n)})$ . In the remainder of this section we show the Guha–Khuller reduction can be used to derive a priority algorithm for set cover from any *memoryless* priority algorithm. Hence we show that one can interpret the argument of Guha and Khuller as follows: If there exists a memoryless priority algorithm for metric facility location with an approximation ratio better than 1.463, then set cover is  $(1 - \varepsilon) \ln n$ -approximable by a priority algorithm for some  $\varepsilon > 0$ , contradicting Theorem 8.

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

**PROOF.** Denote by  $A_F$  the memoryless priority facility location algorithm used in the reduction and by  $A_S(A_F)$  the corresponding set cover algorithm (we use this notation to emphasize precisely which facility location algorithm the reduction employs). Observe that  $A_S(A_F)$  is not necessarily a priority algorithm: this is due to the fact that  $A_S(A_F)$  may consider (and reject) a certain set more than once throughout its execution, before it finally accepts it. However, we will show that there exists a priority set cover algorithm with the same output as  $A_S(A_F)$ .

As can be seen in Figure 2,  $A_S(A_F)$  consists of several “stages”; in each stage  $A_F$  is executed for an opening cost which is uniform for this stage. We can assume, from Lemma 7,

and without loss of generality, that the optimal cost  $l$  is known to the set cover priority algorithm (we remind the reader that this is because Theorem 8 holds even in the case where the optimal set cover cost is known to the algorithm). For a given stage of  $A_S(A_F)$ ,  $A_F$  returns a set  $\mathcal{F}'$  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.

LEMMA 10. *There exists a facility location algorithm  $A'_F$  with the following properties: (i)  $A'_F$  opens the same facilities as  $A_F$ ; (ii) if  $A'_F$  considers a facility that it does not open, none of the remaining (i.e., not yet considered) facilities will be opened; and (iii)  $A_S(A'_F)$  and  $A_S(A_F)$  accept the same collection of sets.*

PROOF. We show how to simulate (iteration-by-iteration)  $A_F$  by an algorithm  $A'_F$  with the desired properties. More precisely, every time  $A_F$  would open a facility,  $A'_F$  will also consider and open the facility in question. If, however,  $A_F$  would not open a certain facility, then  $A'_F$  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 was not opened by  $A_F$ , will continue to receive lowest priority and never be opened in subsequent iterations of  $A'_F$ . This is easy to do since  $A_F$  is a memoryless algorithm: as  $A'_F$  starts a new iteration, we can reconstruct the order in which  $A_F$  has opened facilities before this iteration, say in the order  $F_1, \dots, F_r$ . We consider each of the configurations  $\emptyset, \{F_1\}, \{F_1, F_2\}, \dots, \{F_1, \dots, F_{r-1}\}$ . Then in determining the priority of any facility  $F$ , we consider its priority given each of these configurations. For example, in configuration  $\{F_1, \dots, F_{i-1}\}$ , if  $A_F$  would have given  $F$  higher priority than the facility  $F_i$  that was next opened but would not have opened  $F$ , then  $A'_F$  will give lowest priority to facility  $F$ .

Given the simulation above, it is straightforward that  $A_S(A'_F)$  and  $A_S(A_F)$  accept the same collection of sets.  $\square$

LEMMA 11. *There exists a priority set cover algorithm with the same output as  $A_S(A'_F)$ .*

PROOF. We show how to simulate  $A_S(A'_F)$  by a priority algorithm. For definiteness, think of the priority of a facility being given by a non-negative real number, with priority zero indicating that  $A'_F$  should not consider the facility. The priority of a set  $S_i$  is given by a pair  $(a_i, b_i)$  where  $a_i$  is the priority that  $A'_F$  would give to facility  $f_i$  (namely, the facility to which  $S_i$  corresponds in  $A_S(A'_F)$ ) if  $A_S(A'_F)$  were not beginning a new stage and  $b_i$  is the priority that would be given to facility  $f_i$  if  $A_S(A'_F)$  were beginning a new stage. 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  $b < b'$ . Note that the resulting set cover algorithm does not consider a set it would not open, and is in fact a priority algorithm.  $\square$

Theorem 9 follows directly from Lemmas 10 and 11.  $\square$

**4. 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  $\frac{4}{3}$  for FIXED PRIORITY (but not necessarily GREEDY) metric facility location algorithms with unbounded memory.

**THEOREM 12.** *The approximation ratio of every FIXED PRIORITY GREEDY algorithm for metric facility location is at least  $3 - \varepsilon$ , for arbitrarily small  $\varepsilon$ .*

**PROOF.** Consider the following instance  $(\mathcal{F}, \mathcal{C})$  of the metric facility location problem. Let  $C_1, C_2, \dots, C_d$  be a partition of the  $n$  cities into  $d$  (disjoint) sets of size  $n/d$  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 with respect to  $C_k \cup C_l$  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  $F_{k,l}$  (resp.  $\tilde{F}_{k,l}$ ) to a facility in  $F_{k,l}$  (resp.  $\tilde{F}_{k,l}$ ) 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  $2 - \varepsilon$ . We emphasize that all connection costs are in  $\{1, 3\}$  and that the facilities have uniform opening costs.

Let  $\sigma$  be the ordered sequence of facilities in  $\mathcal{F}$  produced by a FIXED PRIORITY GREEDY algorithm.

**LEMMA 13.** *For every pair  $k, l$ , with  $1 \leq k, l \leq d$ , and  $k \neq l$ , there exists a set  $S_{k,l}$  of  $n/d$  pairs of facilities  $(f_1, \tilde{f}_1), \dots, (f_{n/d}, \tilde{f}_{n/d}) \in (F_{k,l} \times \tilde{F}_{k,l}) \cup (\tilde{F}_{k,l} \times F_{k,l})$  such that: For every  $m$  with  $1 \leq m \leq n/d$ , the facilities  $f_m$  and  $\tilde{f}_m$  are complementary with respect to  $C_k \cup C_l$ ,  $f_m$  precedes  $\tilde{f}_m$  in  $\sigma$ , 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  $m' \neq m$ .
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  $\sigma$ . Let  $S_{k,l}$  be the set of  $n/d$  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 with respect to  $C_k \cup C_l$ , 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  $\sigma$ . Let  $S_{k,l}$  be the set of  $n/d$  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 with respect to  $C_k \cup C_l$ , and  $\tilde{f}_{ij'}^{k,l}$  covers only  $i$  among the cities in  $C_k$ .  $\square$

EXAMPLE. To illustrate the significance of Lemma 13 as well as the intuition behind the partition of the cities into a large number of sets, consider the case  $d = 2$ . Here  $C_1$  and  $C_2$  consist of  $n/2$  cities each. Lemma 13 states that for every ordering  $\sigma$  there exists a set of  $n/2$  pairs of complementary facilities  $(f_1, \bar{f}_1), \dots, (f_{n/2}, \bar{f}_{n/2})$  with the following property: for every  $i \in [n/2]$ ,  $f_i$  precedes  $\bar{f}_i$  in  $\sigma$ , and  $f_i$  covers a city which is not covered by any facility in the set  $\{f_1, \dots, f_{i-1}, f_{i+1}, \dots, f_{n/2}\}$ . Consider an adversary which removes  $\bar{f}_i$  every time the algorithm considers a facility  $f_i$  unless every other facility  $f_j$  has already been considered. Because of the choice of the facility and connection costs, and from the greedy criterion, the algorithm will always open such a facility. Therefore, the algorithm will have to pay the cost of opening  $n/2 - 1$   $f_i$  facilities; at the same time the optimal algorithm will open only the two complementary facilities. It is easy to see the previous argument results in a (weaker) lower bound of  $2 - \varepsilon$ . This is due to the fact that in the case  $d = 2$  half of the cities are covered by a single facility, and hence do not contribute to the algorithm's cost. The intuition behind a partition for large values of  $d$  is that only a few cities, namely,  $(n \log d)/d$ , will be "cheap" for the algorithm, as will become evident in what follows.

Note that for every fixed pair  $k, l$ , either case 1 or case 2 of Lemma 13 (or possibly both) apply. For the purposes of our proof we 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  $C_l \rightarrow C_k$  (resp.  $C_k \rightarrow C_l$ ) to denote that case 1 (resp. case 2) applies.

Define a digraph  $G = (V, E)$  on  $d$  vertices  $v_1, \dots, v_d$  as follows: the directed edge  $(v_k, v_l)$ , with  $k \neq l$  is in  $E$  iff  $C_k \rightarrow C_l$ . From Lemma 13, 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 14. *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  $(v_k, v_l) \in E$ .*

PROOF. See the Appendix.

Lemma 14 implies that there exists a collection  $D$  of at most  $\log d$  sets  $C_{k_1}, \dots, C_{k_p}$  such that for every set  $C_l \notin D$ , there exists  $C_{k_i} \in D$  such that  $C_{k_i} \rightarrow C_l$ . We now describe how the adversary constructs the input that is presented to the algorithm. Recall from Lemma 13 that the set  $S_{k_i, l}$  of facilities satisfies  $C_{k_i} \rightarrow C_l$ . 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  $C_{k_j} \rightarrow C_l$ , and  $C_{k_j} \in D$ . Let  $I = \bigcup \{I_l \mid C_l \notin D\}$ . 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 with respect to  $C_{k_i} \cup C_l$ , except for the case when  $f$  and its complement are the last pair of complementary facilities with respect to  $C_{k_i} \cup C_l$ . Note that from Lemma 13,  $f$ 's complement with respect to  $C_{k_i} \cup C_l$  follows  $f$  in  $\sigma$ , 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 13, 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  $3 - \varepsilon$  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  $(n/d)(d - |D|) \geq (n/d)(d - \log d)$  cities in  $\mathcal{C} \setminus D$  (where  $|D|$  is the size of  $D$ ), we get

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

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

$$\text{cost}(OPT) \leq 2 \cdot (2 - \varepsilon) \cdot (d - |D|)d + n \leq 2 \cdot (2 - \varepsilon) \cdot d^2 + n.$$

Observe that the ratio  $\text{cost}(ALG)/\text{cost}(OPT)$  can be made arbitrarily close to 3, for large, albeit constant,  $d$ .  $\square$

Using an argument along the lines of the proof of Theorem 12 one can derive the following.

**THEOREM 15.** *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  $\Omega(n)$ , 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  $\Omega(n)$  facilities (resp. sets), while the optimal algorithm opens only a constant number of facilities (resp. sets).  $\square$

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  $\frac{4}{3}$  bound shown in Theorem 1 (and which applies to priority algorithms with unbounded memory).

**THEOREM 16.** *The approximation ratio of every FIXED PRIORITY algorithm for metric facility location is at least  $1.366 - \varepsilon$ , for arbitrarily small  $\varepsilon$ .*

PROOF. 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  $2 - \varepsilon$  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 with respect to  $C_{k_i} \cup C_l$  (for some appropriate  $k_i$ ) except for the case  $f$  and its complement are the last pair of complementary facilities with respect to  $C_{k_i} \cup C_l$ . In the latter case we call  $f$  *critical with respect to  $C_l$* , because the adversary's decision on whether to remove  $f$ 's complement with respect to  $C_{k_i} \cup C_l$  is made according to the following rule. Let  $x_l$  be the fraction of the  $|C_l| = n/d$  facilities in  $S_{k_i, l}$  that the algorithm opened immediately after  $f$  is considered. Then the algorithm will not remove  $f$ 's complement with respect to  $C_{k_i} \cup C_l$  if and only if

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

Let us give some intuition about the actions of the adversary. The numerator of the left-hand side of (1) is the cost that the algorithm will pay to accommodate cities in  $C_l$ , in the case when the complement with respect to  $C_{k_i} \cup C_l$  of the critical facility with respect to  $C_l$  is removed; in such a case, the corresponding optimal cost is given by the denominator. On the other hand, the cost of the algorithm if the said facility is not removed, is precisely the numerator of the right-hand side of (1) (and, likewise, the denominator 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  $C_{\text{in}}$ ,  $C_{\text{out}}$ .  $C_{\text{in}}$  contains all  $C_l \notin D$  for which the last critical complement (with respect to  $C_{k_i} \cup C_l$  for some appropriate  $i$ ) in  $I_l$  was not removed, while  $C_{\text{out}}$  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:

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

On the other hand, the optimal cost is as follows:

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

where the term  $3(n/d) \log d$  upper-bounds the connection cost of cities in  $D$ . To facilitate the exposition we can ignore this term, since, for any arbitrarily small constant  $\varepsilon > 0$ , we can find sufficiently large  $d$  such that the term is smaller than  $\varepsilon \cdot n$  (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 that

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

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

**5. 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  $3 - \varepsilon$  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, uniform facility costs, and all instances are unweighted. Can we improve the results by considering other metric distances, non-uniform opening costs, 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?

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. However, can we derive better results for this variant? As in the set cover problem, it seems appropriate to consider only 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.

Finally, it should be noted that while the lower bounds in this paper apply to *disjoint* facility location only, it is possible to prove lower bounds for the complete facility location problem (e.g., by employing the techniques of Davis and Impagliazzo [8]). Specific lower bounds are given in [2].

**Acknowledgements.** 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.

## Appendix. Proofs of Some Lemmas

**PROOF OF LEMMA 7.** Given Lemma 6, it suffices to show that  $\text{cost}(OPT) \geq l$ , since it follows that  $\text{cost}(OPT) = l$ . We can assume that  $n = N(k, l)$ , since the lower-bound construction in [24] 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 [24]) is *exactly*  $\lceil n/l \rceil$  (this follows from (4), (5), and (11))

in [24]). Suppose, by way of contradiction, that  $\text{cost}(OPT) < l$ . Since  $\lceil n/l \rceil$  is an upper bound on the largest size of a set in the instance (as constructed by the adversary), this would imply  $\lceil n/\lceil n/l \rceil \rceil \leq l - 1$ , which implies that  $nl/(n + l) \leq l - 1$ , or equivalently  $n \leq l^2 - l$ . However,  $n$  grows exponentially with  $l$  (see (36) in [24]), which leads to a contradiction for sufficiently large  $n$ .  $\square$

PROOF OF LEMMA 14. 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  $\lceil \binom{d}{2}/d \rceil = \lceil (d - 1)/2 \rceil$ . 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  $d - 1 - \lceil (d - 1)/2 \rceil \leq (d - 1)/2$  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  $V \setminus V'$ .  $\square$

## References

- [1] S. Angelopoulos. Randomized priority algorithms. In *Proceedings of the 1st International Workshop on Approximation and Online Algorithms*, pages 27–40, 2003.
- [2] S. Angelopoulos. Ordering-preserving transformations and greedy-like algorithms. To appear in 2nd Workshop on Approximation and Online Algorithms, September 2004.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] A. Borodin, M. Nielsen, and C. Rackoff. (Incremental) priority algorithms. *Algorithmica*, 37:295–326, 2003.
- [7] V. Chvátal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.
- [8] S. Davis and R. Impagliazzo. Models of greedy algorithms for graph problems. In *Proceedings of the 15th ACM–SIAM Symposium on Discrete Algorithms*, pages 381–390, 2004.
- [9] U. Feige. A threshold of  $\ln n$  for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
- [10] 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.
- [11] S. Guha, N. Mishra, R. Motwani, and L. O’Callaghan. Clustering data streams. In *Proceedings of the 41st Annual IEEE Conference on Foundations of Computer Science*, pages 359–366, 2000.
- [12] D. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22:148–162, 1982.
- [13] 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.
- [14] K. Jain and V.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.
- [15] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.
- [16] 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.

- [17] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] A. Meyerson. Online facility location. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.
- [22] 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-Verlag, Berlin, 2000.
- [23] 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.
- [24] P. Slavík. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25:237–254, 1997.