Foundations of Software Technology and Theoretical Computer Science (Bangalore) 2008.
Editors: R. Hariharan, M. Mukund, V. Vinay; pp -

All-Norms and All-L_p-Norms Approximation Algorithms

Daniel Golovin[1*], **Anupam Gupta**[1†], **Amit Kumar**[2‡], **Kanat Tangwongsan**[1†]

[1] Computer Science Department
Carnegie Mellon University, Pittsburgh PA, USA 15213.

[2] Department of Computer Science & Engineering
Indian Institute of Technology, Hauz Khas, New Delhi, India 110016.

ABSTRACT. In many optimization problems, a solution can be viewed as ascribing a "cost" to each client, and the goal is to optimize some aggregation of the per-client costs. We often optimize some L_p-norm (or some other symmetric convex function or norm) of the vector of costs—though different applications may suggest different norms to use. Ideally, we could obtain a solution that optimizes several norms simultaneously. In this paper, we examine approximation algorithms that simultaneously perform well on all norms, or on all L_p norms.

A natural problem in this framework is the L_p Set Cover problem, which generalizes SET COVER and MIN-SUM SET COVER. We show that the greedy algorithm *simultaneously gives a* $(p + \ln p + O(1))$*-approximation for all* p*, and show that this approximation ratio is optimal up to constants* under reasonable complexity-theoretic assumptions.

We additionally show how to use our analysis techniques to give similar results for the more general *submodular set cover*, and prove some results for the so-called *pipelined set cover* problem. We then go on to examine approximation algorithms in the "all-norms" and the "all-L_p-norms" frameworks more broadly, and present algorithms and structural results for other problems such as k-facility-location, TSP, and average flow-time minimization, extending and unifying previously known results.

1 Introduction

When the solution to an optimization problem affects multiple people or organizations, there is often a trade-off between various efficiency and fairness measures. Typically, there is an abstract "cost" associated with each participant and the objective function is some aggregation of the individual costs. The method of aggregation represents our relative priorities concerning efficiency and fairness. E.g., in *k-median*, given demand points $D \subseteq V$ in a metric space (V, d), we must select k facilities to open: the cost associated with each participant $d \in D$ is its distance to the nearest open facility. Each solution thus induces a cost vector $\mathbf{C} \in \mathbb{R}_+^{|D|}$, and the objective is to minimize $\|\mathbf{C}\|_1 = \sum_{d \in D} \mathbf{C}_d$, the sum of the participant costs: hence, this method of aggregation favors global efficiency over fairness. Another extreme is *k-center*, where we minimize the fairer objective function $\|\mathbf{C}\|_\infty$, the maximum participant cost. Other examples where such trade-offs appear include:

*Supported in part by NSF ITR grants CCR-0122581 (The Aladdin Center) and IIS-0121678

†Supported in part by an NSF CAREER awards CCF-0448095 and CCF-0729022, and by an Alfred P. Sloan Fellowship.

‡Part of this work done while visiting Max-Planck-Institut für Informatik, Saarbrücken, Germany.

- *Sequencing problems*: $\mathbf{C}$ measures the "time" of service for each participant, for example the cover times of the elements in a set cover instance, or the times to reach the vertices in a TSP instance.
- *Scheduling problems*: $\mathbf{C}$ could be the load of the machines or the flow-times of the individual jobs.
- *Allocation problems*: $\mathbf{C}$ measures the quality of service of each participant, for example congestion or dilation in routing problems, and distances in facility location problems.

In general, there are many aggregation functions we might wish to consider. However, if we are feeling particularly ambitious, we might ask if we can efficiently find solutions that *simultaneously* approximate the optimal solutions for each member of a large class of aggregation functions. Formally, we are given a minimization problem and a class of aggregation functions $\mathcal{F}$. For each $f \in \mathcal{F}$, let $\mathbf{C}^*_f$ be the feasible vector minimizing $f(\cdot)$. Then for as small an α as possible, we want to find a feasible cost vector $\mathbf{C}$ such that $f(\mathbf{C}) \leq \alpha \cdot f(\mathbf{C}^*_f)$ for all $f \in \mathcal{F}$. Such a vector $\mathbf{C}$ is a *simultaneous α approximation* for $\mathcal{F}$.

In this paper, we will consider two classes of aggregation functions: the class of *Minkowski L_p norms* $\{L_p \mid p \in \mathbb{R}_{\geq 1}\} \cup \{L_\infty\}$ (i.e., All L_p Norm results), and the class of *all symmetric norms* (i.e., AllNorm results). The L_p norm of $\mathbf{C}$, which is $\|\mathbf{C}\|_p := (\sum_i \mathbf{C}_i^p)^{1/p}$ for a real value $1 \leq p < \infty$ and $\max_i \mathbf{C}_i$ for $p = \infty$, provides a nicely parameterized way of quantifying the efficiency/fairness trade-off.

The question of all-norm minimization was investigated by Kleinberg et al. [KRT01] in their study of *fair* resource allocation algorithms for routing and load balancing, and the problem of all L_p-norms minimization was considered by Azar et al. [AERW04] for machine scheduling. Subsequent work on these topics was done in the papers [KK00, GMP01, GM06]—the concepts studied here are closely linked to *submajorization* of vectors [HLP88], which is even stronger than simultaneously approximating all symmetric norms (and hence all Minkowski norms), see [GM06] for details and many interesting results derived therefrom. For the comprehensive treatment of submajorization and AllNorm approximation, see books by Hardy et al. [HLP88] or Steele [Ste04].

1.1 All L_p-norms Set Cover

The classical set cover problem wants to pick a small number of sets one-by-one to cover the elements *early in the worst-case*, whereas the *min-sum* set cover problem tries to pick the sets to cover the elements *early "on average"*. In this paper, the first question we consider is how to pick sets so that the second (or higher) moments are small: this is just the L_p-Set Cover (L_p SC) problem. We show that the greedy algorithm is, in fact, a $(p + \ln p + 3)$-approximation for all L_p norms *simultaneously!* Moreover, for any fixed p, we cannot hope to do much better using any other algorithm, and hence greedy is essentially the best.

Formally, a set cover instance consists of a ground set $\mathcal{U}$ of n elements, a collection $\mathcal{F}$ of subsets of $\mathcal{U}$, and a cost function $c : \mathcal{F} \to \mathbb{R}_+$. An algorithm picks sets $S_1, S_2, \ldots, S_t$ (in that order) so that their union $\cup_i S_i$ is $\mathcal{U}$. On this ordering, let c_i be the cost of the set S_i; i.e., $c_i = c(S_i)$. Informally, we may think of S_i as corresponding to an action a_i that covers the elements of S_i, and c_i is the time required to execute a_i. Let the *cover index* of an element $e \in \mathcal{U}$ be defined as $index(e) = \min\{i : e \in S_i\}$; i.e., the position of the first set that contains e. The *cover time* of an element $e \in \mathcal{U}$ is defined to be the time required to cover e if we execute actions in this order: i.e., $time(e) = \sum_{i=1}^{index(e)} c_i$. Note that for the case of unit costs, the cover

index and cover time are the same. Given the sequence of sets that the algorithm picks, we obtain a *cover time vector* $\mathbf{C} \in \mathbb{R}_+^n$, where $\mathbf{C}_e$ is the cover time of the element $e \in \mathcal{U}$. The L_p set cover problem is then to find the ordering that minimizes $\|\mathbf{C}\|_p$. It is easy to see that using the L_1 norm and unit costs we obtain the MIN-SUM SET COVER problem [FLT04], whereas using the L_∞ norm we obtain the classical set cover problem [Chv79, Lov75, Joh74].

We prove the greedy algorithm achieves an approximation ratio of $(1+o(1))\min\{p, \ln n\}$ for L_p set cover (which is simultaneously optimal for all L_p norms), and also an $O(\log n)$-approximation in the AllNorm model. Moreover, even if we focus on any fixed value of p, we show that it is impossible to approximate the L_p set cover problem better than $\Omega(p)$ unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$. This lower bound holds for all functions $p(n)$ such that $1 \le p(n) \le \frac{1-\epsilon}{2}\ln(n)$ for all n. We also show that the greedy algorithm achieves an $(p + \ln p + 3)$-approximation in the L_p *Submodular Set Cover problem*, which is a generalization of the L_p set-cover problem to arbitrary submodular functions.

To the best of our knowledge, there has not been any prior work on All L_p Norm approximation for Set Covering problems seeking to minimize all $\|\mathbf{C}\|_p$; of course, there is much work for special values of p. For the classical MINIMUM SET COVER problem (minimize $\|\mathbf{C}\|_\infty$), an $(1+o(1))\ln n$-approximation is known both by greedy and by LP rounding [Joh74, Lov75, Chv79, Sla97, Sri99]. Moreover, one cannot get an $(1-\epsilon)\ln n$-approximation unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$ [Fei98]. For the MIN-SUM SET COVER problem (minimize $\|\mathbf{C}\|_1$), we know that greedy is an optimal 4-approximation [FLT04] (see also [BNBH+98, CFK03]).

1.2 Overview of our Other Results and Related Work

Pipelined Set Cover: This problem was studied in the All L_p Norm framework by Munagala et al. [MBMW05], and seeks to minimize $\|\mathbf{R}\|_p$ where $\mathbf{R}_i$ is the number of *uncovered* elements before the i^{th} set is chosen. To put this in context, the L_1 norm for this problem is the MIN-SUM SET COVER problem, and the L_∞ norm is just $|\mathcal{U}|$. Munagala et al. show that the output of the greedy algorithm is simultaneously a $9^{1/p}$-approximation for the L_p norm, and also give a local-search algorithm that is a $4^{1/p}$ approximation. We show how our proof ideas from MIN-SUM SET COVER give an $(1 + \frac{\ln p}{p} + \frac{3}{p})$-approximation guarantee for the greedy algorithm for this problem; while slightly worse than the previous known guarantee (note $1 + \frac{\ln(4)}{p} \le 4^{1/p} \le 1 + \frac{3}{p}$ for all $p \ge 1$), it extends to the case of *non-uniform costs* where no guarantee was known for the greedy algorithm.

Norm Sampling: We consider the problem of finding a good representative set for the class of all L_p norms with $p \in \mathbb{R}_{\ge 1} \cup \{\infty\}$—namely a set $S \subset \mathbb{R}_{\ge 1} \cup \{\infty\}$ such that an simultaneous α-approximation for all L_p norms with $p \in S$ implies a simultaneous $O(\alpha)$-approximation for all L_p norms with $p \in \mathbb{R}_{\ge 1} \cup \{\infty\}$. This leads us to a notion of *norm sampling*, and we give tight bounds for the size of S necessary and sufficient to well represent (various subsets of) the L_p norms, as well as explicit constructions of such sets.

Facility Location Problems: We return to the example at the beginning of the introduction, where we seek to open k facilities to minimize $\|\mathbf{C}\|_p$, where $\mathbf{C}$ is the vector of assignment costs of demands. It is known that one can get $O(1)$-approximation algorithm for all norms provided we open $O(k\log n)$ facilities [KK00, GM06], and such a $O(\log n)$ blow-up in the

number of open facilities cannot be avoided [KK00]. In contrast, we use the above norm-sampling ideas to give an $O(1)$-approximation algorithm for all L_p norms with *integer values of* p provided we open $O(k\sqrt{\log n})$ facilities, and show that opening $\Omega(k \cdot (\log_k n)^{1/3})$ facilities may be necessary in some instances.

Results via Partial Covering: For sequencing problems such as TSP, where the cost vector is the time to reach each of the n vertices in some graph, or sequencing versions of covering problems (of which L_p set cover is a good example), we show how to use partial covering results to generate AllNorm approximations. For example, we give an AllNorm 16-approximation result for the TSP by drawing on the elegant techniques of Blum et al. [BCC+94] and the large body of subsequent and related work. To extend the result to other problems (like vertex cover and Multicut on trees), we use results from the well-studied area of *partial* covering problems, and the papers of [GKS04, KPS06] in particular.

Flow-Time Scheduling: Some scheduling problems naturally lend themselves to a job-centric perspective. We consider scheduling jobs on parallel machines and look at the vector of flow times for each job: given ε-factor extra speed for each machine, we get an $O(1/\varepsilon^{O(1)})$- approximation algorithms for all norms. This extends previous work of Chekuri et al. [CGKK04] (who proved the result for all L_p norms), Bansal and Pruhs [BP03] (who gave an All L_p Norm result for a single machine). Related work includes results in the machine-centric model (see, e.g., [AERW04, GM06, AT04, AE05]).

1.3 Preliminaries and Notations

A *norm* $\|\cdot\|$ on vectors of length n is a function from $\mathbb{R}^n \to \mathbb{R}$ that satisfies the following: $\|\alpha X\| = |\alpha| \|X\|$ for any $\alpha \in \mathbb{R}$ and $X \in \mathbb{R}^n$, and secondly $\|X + Y\| \leq \|X\| + \|Y\|$ for $X, Y \in \mathbb{R}^n$. The Minkowski L_p norm of X is $\|X\|_p = (\sum_i X_i^p)^{1/p}$ for a real value $1 \leq p < \infty$; the L_∞ norm is just $\|X\|_\infty = \max_i X_i$. It is well-known that for all $X \in \mathbb{R}^n$ and $p < q$, $\|X\|_p \geq \|X\|_q$ [HLP88].

All of the problems we consider in this paper have the property that a solution to the problem induces a vector of length n; thus, for each instance $\mathcal{I}$ of such a problem, we have a set $V(\mathcal{I})$ consisting of all vectors that are induced by some feasible solution to the instance. For a norm $\|\cdot\|$, let $\|X\|$ denote the norm of the vector X. We state two well-known facts for easy reference: the latter follows directly from the convexity of x^p.

Fact 1 (Generalized AM-GM [Ste04]) $\frac{1}{p}A + \frac{p-1}{p}B \geq A^{1/p}B^{(p-1)/p}$

Fact 2 (The Discrete Differential) *Let $p \geq 1$. If the real numbers a, b, and c satisfy $c = a - b \geq 0$, then $a^p - b^p \leq c \cdot p \cdot a^{p-1}$.*

2 The L_p Set Cover Problem

We show that the greedy algorithm simultaneously gives an $(p + \ln p + 3)$-approximation for the L_p Set Cover problem for all p, hence generalizing the fact that it is an $O(\log n)$-approximation for MIN SET COVER (i.e., the $L_\infty \approx L_{\log n}$ case) and 4-approximation for the L_1 case. We then show that for any p, we give a hardness of approximation result of $\Omega(p)$.

2.1 An Upper Bound for the Greedy Algorithm

Consider the familiar setup. We have a universe $\mathcal{U}$ of n elements and a family $\mathcal{F}$ of subsets of $\mathcal{U}$. The greedy algorithm picks sets $S_1, S_2, \ldots, S_t$ from $\mathcal{F}$ until $\cup_i S_i = \mathcal{U}$, such that each S_i satisfies $|S_i \setminus (\cup_{j<i} S_j)| = \max_{S \in \mathcal{F}} \{|S \setminus (\cup_{j<i} S_j)|\}$.

Let c_i be the cost of the set S_i. Let s_i be the cumulative cost of the first i sets picked by the greedy algorithm. That is, $s_0 = 0$ and $s_{i+1} = s_i + c_{i+1}$. Let $X_i = S_i \setminus (\cup_{j<i} S_j)$ be the set of elements with cover index i. Let $R_i = \mathcal{U} - \bigcup_{j=1}^{i-1} X_j$ be the elements uncovered just before the i^{th} set is picked. We use S_i^*, c_i^*, s_i^*, X_i^* and R_i^* to denote the analogous quantities for the optimal algorithm.

For a fixed value of p, the cost of the greedy algorithm (denoted by $\underline{\text{greedy}}$) can be written in terms of the values X_i and R_i as follows:

$$\underline{\text{greedy}} = \left(\textstyle\sum_{i>0} s_i^p \, |X_i|\right)^{1/p} \tag{1}$$

$$= \left(\textstyle\sum_{i>0} (s_i^p - s_{i-1}^p)|R_i|\right)^{1/p}, \tag{2}$$

where the second expression follows from the fact that $|R_{i+1}| = |R_i| - |X_i|$. The cost of the optimal algorithm can be expressed in a similar fashion.

The following lemma upper bounds the cost of greedy by a somewhat exotic expression, which will later turn out to be crucial to our analysis.

Lemma 3 (Upper-bound on Greedy)

$$\underline{\text{greedy}}^p \;\le\; (\underline{\text{greedy}}')^p \;\stackrel{def}{=}\; \sum_{i>0} \left(p \cdot c_i \frac{|R_i|}{|X_i|}\right)^p \cdot |X_i|$$

PROOF. Let $A_i = \left(p \cdot c_i \frac{|R_i|}{|X_i|}\right)^p \cdot |X_i|$ be the i^{th} term in the summation above. Taking the i^{th} terms in the expressions (1) and (2) measuring the cost of the greedy algorithm, and raising them to the p^{th} powers, define $B_i = (s_i^p - s_{i-1}^p)\,|R_i|$ and $C_i = s_i^p\,|X_i|$. It follows from Fact 1 that $\frac{1}{p}\,A_i + \frac{p-1}{p}\,C_i \ge A_i^{1/p}\,C_i^{(p-1)/p} = p \cdot c_i \cdot s_i^{p-1}|R_i| \ge B_i$. The last inequality follows from Fact 2 and the observation that $c_i = s_i - s_{i-1}$. Now, rearranging terms, we have that $A_i \ge p\,B_i - (p-1)\,C_i$; summing this over all i and noting that $\sum_i B_i = \sum_i C_i = \underline{\text{greedy}}^p$, we get that $\sum_i \left(p \cdot c_i \frac{|R_i|}{|X_i|}\right)^p \cdot |X_i| = \sum_i A_i \ge p \sum_i B_i - (p-1)\sum_i C_i = \underline{\text{greedy}}^p$, which completes the proof. ∎

Given this upper bound on the cost of the greedy algorithm, we now compare this to the optimal L_p set cover cost. While the structure of the remainder of the proof follows that by Feige et al. [FLT04] for the L_1 case, we need a few new ingredients, most notably obtaining the correct "price" function.

Theorem 4 (L_p Approximation Guarantee) *The greedy algorithm gives a $(1+p)^{1+1/p} \le (p + \ln p + 3)$-approximation for the L_p set cover problem.*

PROOF. Recall that $\underline{\text{greedy}}$ and $\underline{\text{opt}}$ denote the cost of the greedy algorithm and the optimal algorithm, respectively. We show $\underline{\text{opt}}$ graphically as in Figure 1 (left). The horizontal axis is divided into n equal columns, corresponding to the elements of the universe $\mathcal{U}$. The elements are arranged from left to right in the order that the optimal algorithm covers them. The

column corresponding to the element x has height $(s^*_{index^*(x)})^p$. Thus the area under the curve is $\underline{\text{opt}}^p$.

As Lemma 3 shows, $\underline{\text{greedy}}^p$ can be upper-bounded by the expression $(\underline{\text{greedy}}')^p$. The right panel of Figure 1 models the quantity $(\underline{\text{greedy}}')^p$. The diagram has n columns corresponding to the elements of $\mathcal{U}$ appearing from left to right in the order that the greedy algorithm covers them. For each element of X_i, its corresponding column has height $\lceil p \cdot c_i|R_i|/|X_i| \rceil^p$.

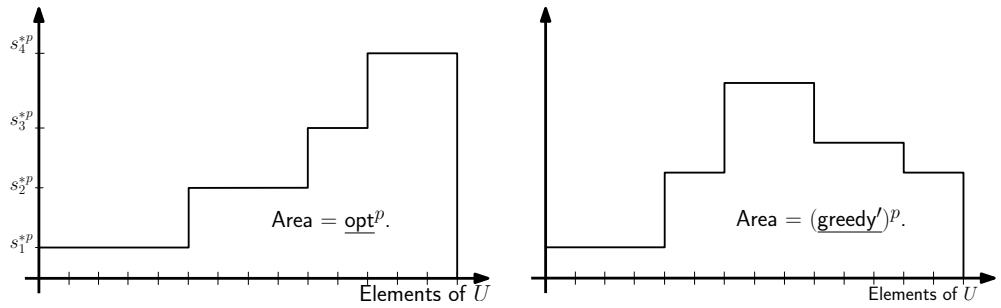

Figure 1: Graphical representations of the cost of the optimal algorithm (left) and an upper bound of the cost of the greedy algorithm (right).

We will now show that the area of the $(\underline{\text{greedy}}')^p$ curve is at most $p^p(1+p)(1+1/p)^p$ times the area of the $\underline{\text{opt}}^p$ curve. To prove this, we scale the $(\underline{\text{greedy}}')^p$ curve down by $\lceil p(1+1/p) \rceil^p$ vertically and by $1+p$ horizontally, and place this scaled curve so that its bottom-right is aligned with the bottom-right of the $\underline{\text{opt}}^p$ curve. Now consider a point $q = (x, y)$ on the original $(\underline{\text{greedy}}')^p$ curve. Suppose the point q corresponds to an element of X_i, so $y \leq \lceil p \cdot c_i|R_i|/|X_i| \rceil^p$. Also the distance to q from the right side is at most $|R_i|$. Therefore, the height of the point q after scaling, which we denote by h, is at most $\left(\frac{1}{1+1/p} \cdot \frac{|R_i|}{|X_i|/c_i}\right)^p$, and the distance from the right (after scaling), denoted by r, is at most $|R_i|/(1+p)$.

In order to show that the point q (after scaling) lies within the $\underline{\text{opt}}^p$ curve, it suffices to show that when the optimal algorithm's cover time is $h^{1/p}$, at least r points remain uncovered. Consider the set R_i. Within this set, the greedy algorithm covers the most elements per unit increase in cover time. Therefore, the number of elements from R_i that the optimal algorithm can cover in time $h^{1/p}$ is at most $\left(\frac{1}{1+1/p} \cdot \frac{|R_i|}{|X_i|/c_i}\right) \frac{|X_i|}{c_i} \leq \frac{1}{1+1/p}|R_i|$, and so at least $\frac{1}{1+p}|R_i|$ elements remain uncovered at time $h^{1/p}$. Since $|R_i|/(1+p) \geq r$, this implies that q (after scaling) lies within the $\underline{\text{opt}}^p$ curve, and hence the scaled-down version of the $(\underline{\text{greedy}}')^p$ curve is completely contained within the $\underline{\text{opt}}^p$ curve. Quantitatively, this implies that $\underline{\text{greedy}}^p \leq (1+p)\lceil p(1+1/p) \rceil^p \ \underline{\text{opt}}^p = (1+p)^{p+1} \ \underline{\text{opt}}^p$. It can be shown that $(1+p)^{1+1/p} \leq p + \ln p + 3$ for $p \geq 1$, which completes the proof. ■

Having shown that the greedy algorithm gives an $O(p)$ approximation for any fixed p, in the full version we give an example for which the greedy algorithm is an $\Omega(p)$ approximation.

Theorem 5 (Tight Example for Greedy) *There is a set system on which greedy yields an $\Omega(p)$ approximation.*

2.2 A Matching Hardness Result for L_p Set Cover

In this section, we show that the greedy algorithm achieves the best possible approximation factor up to constant factors; indeed, we show that even if we fix a value of p, there is no polynomial-time algorithm approximating L_p set cover problem better than $\Omega(p)$ unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$. We first prove a technical lemma.

Lemma 6 *Let $\#OPT(I)$ denote the number of sets an optimal algorithm (for the classical min set cover) needs to cover the set-cover instance I. Let $\epsilon > e^2$. Let $t : \mathbb{N} \to \mathbb{R}_+$ be a non-decreasing function such that $1 \le t(n) \le \log_\epsilon n$ for all n. If there exists an efficient algorithm A such that for all $n > 0$, for all instance I with n elements, A covers at least $n \cdot (1 - \epsilon^{-t(n)})$ elements with $t(n) \cdot \#OPT(I)$ sets, then $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$.*

The proof is standard and can be found in the full version [GGKT07].

Lemma 7 *Suppose $\delta > 0$, and $p(n) = \omega(1)$ is non-decreasing and $1 \le p(n) \le (\frac{1}{2} - \delta)\ln n$ for all n. Then the L_p set-cover problem is $\Omega(p)$-hard to approximate unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$.*

PROOF. Assume $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{O(\log\log n)})$. Let p (the norm parameter), $\epsilon > e^2$ be given, and let $t(n) = p(n)$. (Note that since $t(n)$ must be less than $\log_\epsilon n$, and $\epsilon > e^2$, we need the upper bound of $(\frac{1}{2} - \delta)\ln n$ on $p(n)$.) As a direct consequence of Lemma 6 and our complexity assumption, we know that for all efficient algorithm A, there is $n > 0$ such that there is an instance I of size n such that using $t(n) \cdot \#OPT(I)$ sets, A has at least $n \cdot \epsilon^{-t(n)}$ elements remaining.

Let A be any polynomial-time algorithm for solving L_p set cover. Fix n and such an instance I. Let $\underline{\mathsf{opt}}$ denote the L_p cost of any optimal algorithm on the instance I, and let $\underline{\mathsf{alg}}$ denote the L_p cost of the algorithm A. As before, let X_i denote the elements with cover index i and let R_i denote the elements with cover index i or greater A's solution, and let X_i^* and R_i^* denote the analogous sets for the optimal solution. We know that $\underline{\mathsf{opt}}^p = \sum_{i=1}^{k} i^p |X_i^*| \le n \cdot [\#OPT(I)]^p$, because the classical solution is also a solution of the L_p version. On the other hand, $\underline{\mathsf{alg}}^p \ge s^p \cdot |R_s|$ for all $s > 0$. In particular, with $s = p \cdot \#OPT(I)$ and our lower bound on $|R_s|$ from Lemma 6, we conclude $\underline{\mathsf{alg}}^p \ge (\#OPT(I) \cdot p)^p \cdot \frac{n}{\epsilon^p}$ Therefore, $\underline{\mathsf{alg}}/\underline{\mathsf{opt}} \ge \left((p/\epsilon)^p\right)^{1/p} = p/\epsilon = \Omega(p)$. ∎

Lemma 8 *For $p(n) = O(1)$, it is impossible to approximate L_p set cover better than $\Omega(p)$ unless $\mathbf{P} = \mathbf{NP}$.*

PROOF. Feige et al. [FLT04] shows that, for all $c_0, \epsilon > 0$, there are set cover instances such that it is **NP**-hard to distinguish between the following two cases: (**1**) There is a set cover of size t, or (**2**) For all integers x such that $1 \le x \le c_0 t$, every collection of x sets leaves at least a fraction of $(1 - 1/t)^x - \epsilon$ of the elements uncovered.

It follows that if we guess t, any algorithm leaving fewer than $((1 - 1/t)^x - \epsilon)\, n$ elements uncovered after buying x sets, for any $x \in [1, c_0 t]$, allows us to solve an **NP**-Complete problem. Thus unless $\mathbf{P} = \mathbf{NP}$, every polynomial time algorithm run on these instances has at least $((1 - 1/t)^x - \epsilon)\, n$ elements uncovered after buying x sets, for any $x \in [1, c_0 t]$.

Now fix p and a polynomial time algorithm A and let $\underline{\mathsf{alg}}^p$ be the p^{th} power its cost for the L_p set-cover problem. Let $\underline{\mathsf{opt}}^p$ denote the corresponding quantity for the optimal solution. Let $g(x) := x^p - (x-1)^p$. Recall $\underline{\mathsf{alg}}^p = \sum_x |R_x| \cdot g(x)$, where R_x is the set of elements with cover index at least x. Suppose that there is a set cover of size t. In that case it is not too hard

to show that $\underline{\mathsf{opt}}^p \leq \sum_{x=1}^{t} \left(\frac{n}{t}\right) x^p$, since after buying x sets the optimal solution covers at least $\frac{n}{t}x$ elements. Thus $\underline{\mathsf{opt}}^p \leq n \cdot t^p$. On the other hand:

$$\underline{\mathsf{alg}}^p = \sum_{x \geq 1} |R_x| g(x) \geq \sum_{x=1}^{c_0 \cdot t} \left((1 - \frac{1}{t})^x - \epsilon \right) n \cdot g(x) \approx n \int_{x=1}^{c_0 \cdot t} \left(e^{-\frac{x}{t}} - \epsilon \right) g(x) dx$$

Note that $t = \omega(1)$, so $(1 - 1/t)^x \approx e^{-x/t}$ is an arbitrarily accurate approximation. If we can set $c_0 > (p+1)$ and $\epsilon \leq e^{-(p+1)}/2$ it is not too hard to show $\underline{\mathsf{alg}}^p = \Omega(nt^p \left(\frac{p}{e}\right)^p)$, simply by considering the contribution of $\sum_{x=pt}^{(p+1)\cdot t} \left(e^{-x/t} - \epsilon\right) n \cdot g(x)$ to $\underline{\mathsf{alg}}^p$. Thus $\underline{\mathsf{alg}}^p / \underline{\mathsf{opt}}^p = \Omega((\frac{p}{e})^p)$, and we obtain a gap of $\underline{\mathsf{alg}}/\underline{\mathsf{opt}} = \Omega(p)$ for all constant p. ∎

Combining Lemma 7 and Lemma 8 immediately yields the following theorem.

Theorem 9 *Unless* **NP** $\subseteq$ **DTIME**$(n^{O(\log \log n)})$*, for all* $\delta > 0$ *and* $p = p(n)$ *such that* $1 \leq p(n) \leq (\frac{1}{2} - \delta) \ln(n)$*, it is impossible to approximate* L_p *set cover better than* $\Omega(p)$*.*

2.3 Submodular Set Cover

We now consider a generalization of the L_p set cover problem. Our setting now assumes a (monotone) submodular function $f : 2^V \to \mathbb{R}_+$. Using techniques similar to those above, we can analyze the greedy algorithm's performance on this generalization, and obtain the same approximation guarantee. Thus, if action x_i takes c_i time to perform, and we perform actions $x_1, x_2, \ldots, x_k$ in that order, the total cost will be

$$\left(\sum_{i=1}^{k} \left(f(S_i) - f(S_{i-1}) \right) \cdot \left(\sum_{j=1}^{i} c_j \right)^p \right)^{1/p}$$

where $S_i := \{x_1, x_2, \ldots, x_i\}$. The objective is to select the permutation that minimizes this cost. The proof of the following theorem appears in the full version [GGKT07].

Theorem 10 (Submodular L_p Approximation Guarantee) *The greedy algorithm gives a* $(1 + p)^{1+1/p} \leq p + \ln p + 3$*-approximation for the submodular* L_p *set cover problem.*

2.4 The Pipelined Set Cover Problem

Closely related to the L_p set cover problem is the L_p pipelined set cover problem. In L_p-pipelined set cover, the cost function is given by:

$$\mathsf{cost} = \left(\sum_{i \geq 0} c_i |R_i|^p \right)^{1/p}$$

This formulation follows [MBMW05] but incorporates the notion of cost for each set. [§] When $p = 1$, this cost function is the same that for the L_p case (and the min sum set cover problem). For this problem, we use the technique in the proof of Theorem 4 to argue that the greedy algorithm achieves the following approximation ratio; previous work [MBMW05] gave no approximation guarantee the general costs case. The proof is given in the full version.

Theorem 11 (Pipelined Set-Cover Approximation Guarantee) *The standard greedy algorithm gives a* $(1 + \frac{\ln p}{p} + \frac{3}{p})$*-approximation for the* L_p *pipelined set-cover problem.*

[§]This expression, in fact, differs from that defined by Munagala et al. [MBMW05]: their objective raises c_i to the p^{th} power. However, this only changes the quantity minimized in the greedy step, and hence we use this expression for convenience.

3 All L_p Norm Approximations via Sampling Norms

We now ask the following question: *Is there a small "basis" set of L_p norms that "approximate" all other L_p norms?* Formally, given two vectors X and Y of length n each, is there a set S of indices such that if $\|X\|_p \leq \|Y\|_p$ for all $p \in S$, then the same inequality holds (up to a constant approximation) for *all* L_p norms? Given such a set S, we can imagine finding a solution for each L_p with $p \in S$, and then "composing" them together to get solution that is good for *all* L_p norms. In this section, we will show that there is indeed such a set S of size $O(\log n)$; if we are interested in maintaining L_p norms only for *integer* p, then we can get a set of size $O(\sqrt{\log n})$. Moreover, we show that both these bounds are tight. Proofs omitted from this section appear in the full version [GGKT07].

Definition 12 (α-Sampling) *For a domain $D \subseteq \mathbb{R}_{\geq 1} \cup \{\infty\}$, a set $S \subseteq D$ is an* α-sampling of D of order n *if for all pairs of non-negative vectors $X, Y \in \mathbb{R}^n_{\geq 0}$*

$$\|X\|_p \leq \|Y\|_p \text{ for all } p \in S \quad \Rightarrow \quad \|X\|_p \leq \alpha \cdot \|Y\|_p \text{ for all } p \in D.$$

Such samplings prove useful in the All L_p Norm framework in the following way.

Theorem 13 *Given a minimization problem whose objective function is the L_p norm of some cost vector, and an α-sampling S of $D \subseteq \mathbb{R}_{\geq 1} \cup \{\infty\}$, then a cost vector $\mathbf{C}$ that is a simultaneous β-approximation for the class $\{L_p \mid p \in S\}$ is a simultaneous $\alpha\beta$-approximation for the class $\{L_p \mid p \in D\}$.*

We prove the following tight bounds on the size of $O(1)$-samplings.

Theorem 14 (Tight Bounds on $O(1)$-Samplings) *There exists an $O(1)$-sampling of the domain $D_{reals} = \mathbb{R}_{\geq 1} \cup \{\infty\}$ of order n with size $|S| = O(\log n)$, and an $O(1)$-sampling of the domain $D_{ints} = \mathbb{Z}_{\geq 1} \cup \{\infty\}$ of order n with size $O(\sqrt{\log n})$. Moreover, one cannot obtain smaller $O(1)$-samplings for either of these domains.*

3.1 All L_p Norm Approximations for Facility Location Problems

In this section, we show how the $O(1)$-samplings immediately give algorithms for the All L_p Norm k-facility location problems. As mentioned in the introduction, we can imagine an abstract facility location problem where given a metric space (V, d) with demand points $D \subseteq V$, we open a set of at most k facilities $F \subseteq V$ and assign each demand to a facility. This naturally gives a vector $\mathbf{C}$ of *assignment costs* for the demands with each solution: the k-median problem now minimizes $\|\mathbf{C}\|_1$, the k-means problem looks at $\|\mathbf{C}\|_2$, and the k-center problem at $\|\mathbf{C}\|_\infty$, etc. Let $\underline{\text{opt}}_p(k)$ denote a solution opening k facilities that minimizes the L_p norm of the vector of assignment costs. For any set of open facilities F, let $\mathsf{Cost}_p(F)$ denote the ℓ_p norm of the resulting vector of assignment costs. The following theorem shows how to get an All L_p Norm approximation to such problems.

Theorem 15 *There exists a set F of $O(k \log n)$ facilities F such that $\mathsf{Cost}_p(F) \leq O(1) \cdot \mathsf{Cost}_p(\underline{\text{opt}}_p(k))$ for all $p \geq 1$. If we want this to hold for all L_p norms for integer values of p only, then we need only $O(k\sqrt{\log n})$ facilities. Moreover, we can find these facilities in polynomial time in both cases.*

The proof is immediate from Theorems 13 and 14, and the fact that for any $1 \leq p < \infty$, one can use existing techniques to get an $O(1)$-approximation algorithm for minimizing the ℓ_p norm $\|\mathbf{C}\|_p$. Indeed, all the approximation algorithms for the k-median problem

cited above have the following additional property—if the underlying space only satisfies a λ-relaxed triangle-inequality (i.e., the distances satisfy $d(x,y) \leq \lambda \cdot (d(x,z) + d(y,z)$ for the parameter $\lambda \geq 1$), then these algorithms give an $O(\lambda)$-approximation algorithm for the k-median problem. The problem of minimizing the (p^{th} power of) the ℓ_p norm of assignment cost can be thought of as the k-median problem where distance between two points x and y is given by $d(x,y)^p$. Now these distances satisfy the $\lambda = 2^p$-relaxed triangle-inequality, and hence we get an $[O(2^p)]^{1/p}$-approximation algorithm for the ℓ_p norm.

Kumar and Kleinberg showed that we need to open $\Omega(k \log n)$ facilities to get an $O(1)$-AllNorm-approximation. That proof does not work for the All L_p Norm case; however, we can show the following result.

Theorem 16 *Given a parameter α, there exists a metric space over n demand points such that for a set of facilities F satisfying $\mathsf{Cost}_p(F) \leq \alpha \cdot \underline{\mathsf{opt}}_p(k)$ for all integer $p \geq 1$, $|F| \geq \Omega\big(k\big(\frac{\log n}{\log(\alpha k)}\big)^{\frac{1}{3}}\big)$. In fact, the lower bound holds even for L_p norms with integer p.*

It is an interesting open problem if we can open $o(k \log n)$ facilities and still be $O(1)$-competitive against all L_p norms.

4 AllNorm Approximation Algorithms

In the previous sections, we were interested in All L_p Norm approximations, and situations where focusing on L_p norms (instead of all symmetric norms) would give more nuanced results. In this section, we give results for the AllNorm case; complete proofs of the theorems in this section appear in the full version [GGKT07].

For a vector X, define $\overleftarrow{X}$ as the vector obtained by sorting the coordinates of X in descending order. Given vectors X and Y of length n each, we say that X is *α-submajorized* by Y (written as $X \prec_\alpha Y$) if for all $i \in [n]$, $\sum_{j \leq i} \overleftarrow{X}_j \leq \alpha \sum_{j \leq i} \overleftarrow{Y}_j$ (i.e., the partial sums of $\overleftarrow{X}$ are at most α times the partial sums in $\overleftarrow{Y}$). Intuitively, this means that the k unhappiest elements in X are together at most α times worse off than the k unhappiest elements of Y: we will want to find solutions X which are α-submajorized by *any other* solution Y (for small α). The following result is well-known (see, e.g., [Ste04]).

Theorem 17 *Let X and Y be two vectors of equal length, such that X is α-submajorized by Y. Then $f(X) \leq f(\alpha \cdot Y)$ for all real symmetric convex functions. In particular, if f is a symmetric norm, then $f(X) \leq \alpha f(Y)$.*

4.1 AllNorm Approximation from Partial Covering Algorithms

We now show how solutions for "partial covering" problems can be used to prove submajorization results; these submajorization results immediately lead to AllNorm approximations for these problems by Theorem 17. *Partial covering* problems include the k-MST problem (find a tree of minimum cost spanning at least k nodes), or the k-vertex cover problem (find a set of nodes of minimum size/cost that covers at least k edges). In this paper, we show how an $O(1)$-approximation to the k-MST problem implies an $O(1)$-submajorization result, and how these ideas extend to other partial cover problems.

Theorem 18 *For a TSP instance on a graph $G = (V, E)$, given a tour π, let t_i be the time at which the salesperson reaches vertex v_i, and let $T_\pi = (t_1, t_2, \ldots, t_n)$ be the vector of these* arrival times *sorted in ascending order. Then there is a solution where the arrival time vector is α-submajorized by the corresponding vector in any other solution, where $\alpha \leq 16$.*

The ideas behind this theorem can be used to show that Set Cover problem admits an $O(\log n)$-AllNorm approximation, Vertex Cover an 8-AllNorm approximation, etc. Let us sketch the idea for Vertex Cover: first use the fact that k-vertex cover admits a 2-approximation [BB98, Hoc98, BY01, GKS04]. This gives us an algorithm that given a budget B, finds a solution of cost $2B$ in poly-time which covers at least as many edges as any other solution of cost B. Setting the value of B to be successive powers of 2, we can argue that if any other algorithm covers k elements with cost at most 2^{i-1}, then we would have covered at least k elements with cost at most $4 \cdot 2^i$; this gives us an 8-submajorization. See the papers [GKS04, KPS06] for results on partial covering problems (all of which can be similarly extended).

4.2 AllNorm Algorithms for Flow Time on Parallel Machines

Finally, we consider the problem of scheduling jobs on parallel machines: given a schedule $\mathcal{A}$, the vector of interest is the vector $F^{\mathcal{A}}$ of *flow times*, where the flow time is the difference between its completion time and release date—hence, the ℓ_1 norm of this vector is the problem of minimizing the average flow time on parallel machines: see, [CKZ01] and the references therein for several polynomial-time logarithmic-approximation algorithms.

It is known that for any schedule $\mathcal{A}$, the All L_p Norm-guarantee $\alpha_{ALN}(F^{\mathcal{A}})$ is unbounded even if there is only one machine [BP04]: hence results have been given in the *resource augmentation* framework by giving our machines $(1+\varepsilon)$-speed. In particular, Bansal and Pruhs [BP04], and Chekuri et al. [CGKK04] gave results showing that given any constant $\varepsilon > 0$, we can get an $O(\frac{1}{\varepsilon^{O(1)}})$-approximation algorithm for all ℓ_p norms. In this paper, we show that one can extend their results to a submajorization, and hence AllNorm result.

Theorem 19 *There exists a schedule $\mathcal{A}$ such that $F^{\mathcal{A}}$ β-submajorizes $F^{\mathcal{B}}$ for all schedules $\mathcal{B}$, where β is a constant (depending only on ε).*

References

[AE05] Yossi Azar and Amir Epstein. Convex programming for scheduling unrelated parallel machines. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 331–337, New York, 2005. ACM.

[AERW04] Yossi Azar, Leah Epstein, Yossi Richter, and Gerhard J. Woeginger. All-norm approximation algorithms. *J. Algorithms*, 52(2):120–133, 2004.

[AT04] Yossi Azar and Shai Taub. All-norm approximation for scheduling on identical machines. In *Algorithm theory—SWAT 2004*, volume 3111 of *Lecture Notes in Comput. Sci.*, pages 298–310. Springer, Berlin, 2004.

[BB98] Nader H. Bshouty and Lynn Burroughs. Massaging a linear programming solution to give a 2-approximation for a generalization of the vertex cover problem. In *STACS 98 (Paris, 1998)*, volume 1373 of *Lecture Notes in Comput. Sci.*, pages 298–308. Springer, Berlin, 1998.

[BCC+94] Avrim Blum, Prasad Chalasani, Don Coppersmith, Bill Pulleyblank, Prabhakar Raghavan, and Madhu Sudan. The minimum latency problem. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 163–171. ACM Press, 1994.

[BNBH+98] Amotz Bar-Noy, Mihir Bellare, Magnús M. Halldórsson, Hadas Shachnai, and Tami Tamir. On chromatic sums and distributed resource allocation. *Inform. and Comput.*, 140(2):183–202, 1998.

[BP03] Nikhil Bansal and Kirk Pruhs. Server scheduling in the L_p norm: a rising tide lifts all boat. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 242–250, New York, 2003. ACM.

[BP04] Nikhil Bansal and Kirk Pruhs. Server scheduling in the weighted l_p norm. In *LATIN 2004: Theoretical informatics*, volume 2976 of *Lecture Notes in Comput. Sci.*, pages 434–443. Springer, Berlin, 2004.

[BY01] Reuven Bar-Yehuda. Using homogeneous weights for approximating the partial cover problem. *J. Algorithms*, 39(2):137–144, 2001.

[CFK03] Edith Cohen, Amos Fiat, and Haim Kaplan. Efficient sequences of trials. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003)*, pages 737–746, New York, 2003. ACM.

[CGKK04] Chandra Chekuri, Ashish Goel, Sanjeev Khanna, and Amit Kumar. Multi-processor scheduling to minimize flow time with ϵ resource augmentation. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 363–372, New York, 2004. ACM.

[Chv79] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979.

[CKZ01] Chandra Chekuri, Sanjeev Khanna, and An Zhu. Algorithms for minimizing weighted flow time. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, pages 84–93 (electronic), New York, 2001. ACM.

[Fei98] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[FLT04] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[GGKT07] Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-Norms and All-L_p-Norms approximation algorithms. Technical Report CMU-CS-07-153, School of Computer Science, Carnegie Mellon University, September 2007.

[GKS04] Rajiv Gandhi, Samir Khuller, and Aravind Srinivasan. Approximation algorithms for partial covering problems. *J. Algorithms*, 53(1):55–84, 2004.

[GM06] Ashish Goel and Adam Meyerson. Simultaneous optimization via approximate majorization for concave profits or convex costs. *Algorithmica*, 44(4):301–323, 2006.

[GMP01] Ashish Goel, Adam Meyerson, and Serge Plotkin. Combining fairness with throughput: online routing with multiple objectives. *J. Comput. System Sci.*, 63(1):62–79, 2001.

[HLP88] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge Mathematical Library. Cambridge University Press, Cambridge, 1988. Reprint of the 1952 edition.

[Hoc98] Dorit S. Hochbaum. The t-vertex cover problem: extending the half integrality framework with budget constraints. In *Approximation algorithms for combinatorial optimization (Aalborg, 1998)*, volume 1444 of *Lecture Notes in Comput. Sci.*, pages 111–122. Springer, Berlin, 1998.

[Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[KK00] Amit Kumar and Jon Kleinberg. Fairness measures for resource allocation. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 75–85. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000.

[KPS06] Jochen Könemann, Ojas Parekh, and Danny Segev. A unified approach to approximating partial covering problems. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 468–479, London, UK, 2006. Springer-Verlag.

[KRT01] Jon Kleinberg, Yuval Rabani, and Éva Tardos. Fairness in routing and load balancing. *J. Comput. System Sci.*, 63(1):2–20, 2001. Special issue on internet algorithms.

[Lov75] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13(4):383–390, 1975.

[MBMW05] Kamesh Munagala, Shivnath Babu, Rajeev Motwani, and Jennifer Widom. The pipelined set cover problem. In *Database theory—ICDT 2005*, volume 3363 of *Lecture Notes in Comput. Sci.*, pages 83–98. Springer, Berlin, 2005.

[Sla97] Petr Slavík. A tight analysis of the greedy algorithm for set cover. *J. Algorithms*, 25(2):237–254, 1997.

[Sri99] Aravind Srinivasan. Improved approximation guarantees for packing and covering integer programs. *SIAM J. Comput.*, 29(2):648–670, 1999.

[Ste04] J. Michael Steele. *The Cauchy-Schwarz master class*. MAA Problem Books Series. Mathematical Association of America, Washington, DC, 2004. An introduction to the art of mathematical inequalities.