All-Norms and All-L_p-Norms Approximation Algorithms

Daniel Golovin[1] **Anupam Gupta**[2] **Amit Kumar**[3]
Kanat Tangwongsan[4]

September 20, 2007
CMU-CS-07-153

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by NSF ITR grants CCR-0122581 (The Aladdin Center) and IIS-0121678

[2]Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by an NSF CAREER award CCF-0448095, and by an Alfred P. Sloan Fellowship.

[3]Department of Computer Science & Engineering, Indian Institute of Technology, Hauz Khas, New Delhi, India - 110016. Part of this work was done while the author was at Max-Planck-Institut für Informatik, Saarbrücken, Germany.

[4]Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213. Supported in part by NSF CAREER award CCF-0448095, and by an Alfred P. Sloan Fellowship.

Keywords: set-cover problems, approximation algorithms, combinatorial optimization, sampling minkowski norms

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 optimized 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 to consider in this framework is the set-cover problem: suppose we pick sets $S_1, S_2, \ldots, S_t$ (in that order), and define the *cover time* of element e to be $C_e = \min\{i \mid e \in S_i\}$, the index of the first set that covers e. Minimizing the maximum cover time $\|\mathbf{C}\|_\infty$ gives us the classical MIN SET COVER, for which the greedy algorithm is an $O(\log n)$ approximation (which is best possible). Minimizing the average (or total) cover time $\sum_e C_e = \|\mathbf{C}\|_1$ gives the MIN-SUM SET COVER problem, for which the greedy algorithm gives a 4-approximation (which also is tight). This leads us to a natural question:

> *How well does the greedy algorithm perform for the L_p Set Cover problem, where the objective function is $\|\mathbf{C}\|_p = (\sum_e C_e^p)^{1/p}$ for $1 \leq p \leq \infty$?*

We give tight results for this problem: the greedy algorithm *simultaneously* gives an $O(p)$-approximation for L_p-Set-Cover for all values of $1 \leq p < \infty$ (even for the weighted version). There are simple examples where this is tight for all values of p. In fact, we also show that for every fixed value of p, no efficient algorithm can obtain an approximation guarantee better than $\Omega(p)$ under suitable complexity assumptions. We 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

- *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.

These questions were investigated by Kleinberg et al. [KRT01] in their study of *fair* resource allocation algorithms. Their notion of a fair allocation was one which came close to the max-min fair solution. They considered routing and load balancing problems in this setting. Kumar and Kleinberg [KK00] used a stronger notion of fairness: for cost-minimization problems, a cost vector $\mathbf{C}$ was α-fair (the precise term used was different) if for every $i \leq n$ and $c \in \mathbb{R}$, if there was a feasible solution in which i clients incurred cost at most c, then any set of i clients incurred a cost of at most $\alpha \cdot x$ in our cost vector $\mathbf{C}$. This happens to be the same notion as that of *submajorization* of vectors [HLP88], which is even stronger than simultaneously approximating all symmetric norms. Kumar and Kleinberg [KK00] considered scheduling, facility location and bandwidth allocation problems under this framework. Goel et al. [GMP01] obtained poly-logarithmic-fair algorithms (in this stronger notion of fairness) for routing and bandwidth allocation, and Goel and Meyerson [GM06] showed how to obtain the smallest such α for a variety of problems through linear programming techniques. (For the comprehensive treatment of inequalities, submajorization and AllNorm approximation, one can refer to the book by Hardy et al. [HLP88]; less ambitious readers may want to refer to the book by Steele [Ste04].)

1.1 Overview of Our Results.

Set Cover Problems: In our framework, 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 $c \colon \mathcal{F} \to \mathbb{R}_+$. 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. Formally, the *cover time* of an element $e \in U$ is $time(e) = \sum_{i=1}^{index(e)} c_i$; 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 the L_p set cover (L_p SC) 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 show that the greedy algorithm achieves an $O(\log n)$ approximation in the AllNorm model, and that its approximation ratio of $\Theta(p)$ is in fact *simultaneously* optimal for all L_p norms. Specifically, we show that the classical greedy algorithm achieves an $O(p)$ approximation ratio *for all* $p = p(n)$. Moreover, even if we focus on any fixed value of p, we show that it is impossible to approximate the L_p SC 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 $O(p)$ approximation in the L_p Submodular Set Cover problem, which is a generalization of L_p SC in which an arbitrary submodular function on sets of actions encodes how many elements have been covered.

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 $O(\log n)$ 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]).

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 $|U|$. 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 gives a local-search algorithm that is a $4^{1/p}$ approximation. We show how our proof ideas from MIN-SUM SET COVER give an $O(1+p)^{1/p}$-approximation guarantee for the greedy algorithm for this problem; while worse than the previous known guarantee, it extends to the case of *non-uniform costs* where no guarantee was known for the greedy algorithm.

Structural Results and Norm Sampling: We give a structural result concerning the performance of the greedy algorithm which we have found useful and which may be of independent interest. We also 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 is sometimes required.

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]).

1.2 Preliminaries and Notation

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. For a vector $X \in V(\mathcal{I})$, let $\|X\|_p$ denote the L_p norm $(\sum_i X_i^p)^{1/p}$ of the vector X. We state two well-known facts for easy reference: the latter follows directly from the convexity of x^p.

Fact 1.1 (The Generalized AM-GM Inequality [Ste04]) $\frac{1}{p}A + \frac{p-1}{p}B \geq A^{1/p}B^{(p-1)/p}$

Fact 1.2 (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

2.1 An Upper Bound for the Greedy Algorithm

In this section, we show that the greedy algorithm gives an $O(p)$ approximation to the L_p SC problem. 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 $\mathcal{F}$ so that the union $\cup_i S_i$ is $\mathcal{U}$. 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 $X_i = S_j \backslash (\cup_{j<i} S_j)$ be the set of elements with cover index i. Let $R_i = \mathcal{U} - \bigcup_{j=1}^{i-1} X_i$ 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 $\underline{\mathsf{greedy}}$) can be written in terms of the values X_i and R_i as follows:

$$\begin{aligned} \underline{\mathsf{greedy}} &= \left(\textstyle\sum_{i>0} s_i^p \, |X_i|\right)^{1/p} &\quad (2.1)\\ &= \left(\textstyle\sum_{i>0} (s_i^p - s_{i-1}^p) |R_i|\right)^{1/p}, &\quad (2.2) \end{aligned}$$

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 2.1 (Upper-bound on Greedy Cost)

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

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 (2.1) and (2.2) 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 1.1 that

$$\tfrac{1}{p}\, A_i + \tfrac{p-1}{p}\, C_i \ge A_i^{1/p}\, C_i^{(p-1)/p} = p \cdot c_i \cdot s_i^{p-1}|R_i| \ge B_i.$$

The last inequality follows from Fact 1.2 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 $\sum_i B_i = \sum_i C_i = \underline{\mathsf{greedy}}^p$, we get that

$$\textstyle\sum_i \left(p \cdot c_i \frac{|R_i|}{|X_i|}\right)^p \cdot |X_i| = \sum_i A_i \ge p \sum_i B_i - (p-1) \sum_i C_i = \underline{\mathsf{greedy}}^p,$$

which completes the proof. ■

Given this upper bound on the cost of the greedy algorithm, we now compare this to the optimal L_p SC cost. The basic structure of the remainder of the proof is similar that given by Feige et al. [FLT04] for the L_1 case.

Theorem 2.2 (L_p Approximation Guarantee) *The greedy algorithm gives a $p \cdot (1+p)^{1/p}(1+1/p) \le 4p$-approximation for the L_p SC problem.*

Proof: Recall that $\underline{\mathsf{greedy}}$ and $\underline{\mathsf{opt}}$ denote the cost of the greedy algorithm and the optimal algorithm, respectively. We represent $\underline{\mathsf{opt}}$ 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 clearly equal to $\underline{\mathsf{opt}}^p$.

As Lemma 2.1 shows, $\underline{\mathsf{greedy}}^p$ can be upper-bounded by the expression $(\underline{\mathsf{greedy}'})^p$ that we derived. The right panel of Figure 1 models the quantity $(\underline{\mathsf{greedy}'})^p$. 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 $[p \cdot c_i|R_i|/|X_i|]^p$.

We will now show that the area of the $(\underline{\mathsf{greedy}'})^p$ curve is at most $p^p(1+p)(1+1/p)^p$ times the area of the $\underline{\mathsf{opt}}^p$ curve. To prove this, we scale the $(\underline{\mathsf{greedy}'})^p$ curve down by $[p(1+1/p)]^p$ vertically and by $1+p$ horizontally, and place this scaled curve so that its bottom-right is aligned with the bottom-right of the $\underline{\mathsf{opt}}^p$ curve. Now consider a point $q = (x, y)$ on the original $(\underline{\mathsf{greedy}'})^p$ curve. Suppose the point q corresponds to an element of X_i, so $y \le [p \cdot c_i|R_i|/|X_i|]^p$. 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 $\underline{\mathsf{opt}}^p$ 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

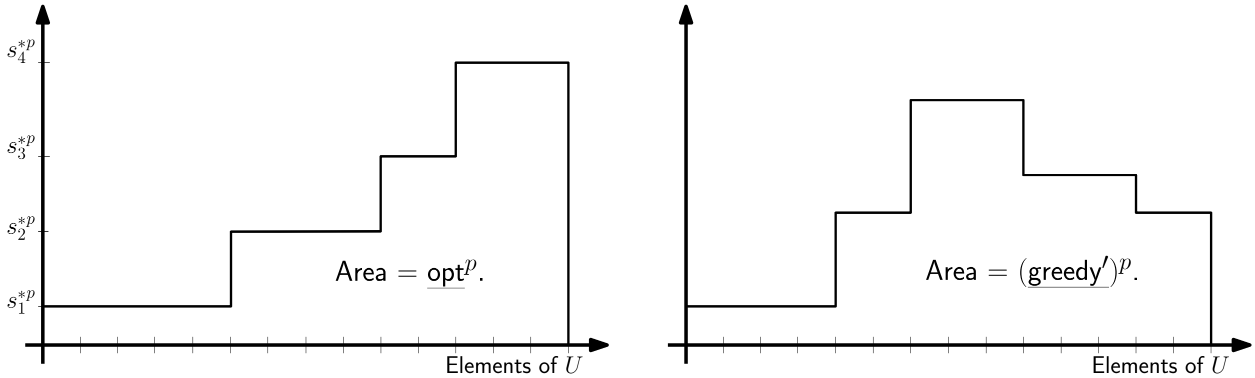

Figure 1: Graphical representations of the cost of the optimal algorithm (left) and an upper bound of the cost of the greedy algorithm (right).

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 $\underline{\mathsf{opt}}^p$ curve, and hence the scaled-down version of the $(\underline{\mathsf{greedy}}')^p$ curve is completely contained within the $\underline{\mathsf{opt}}^p$ curve. Quantitatively, this implies that $\underline{\mathsf{greedy}}^p \leq (1+p)[p(1+1/p)]^p \; \underline{\mathsf{opt}}^p$, completing the proof. ■

Having shown that the greedy algorithm gives an $O(p)$ approximation for any fixed p, it is not hard to give an example for which the greedy algorithm is an $\Omega(p)$ approximation; the proof appears in the appendix.

Theorem 2.3 (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 is 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 2.4 *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 $t\colon \mathbb{N} \to \mathbb{R}^+$ be a non-decreasing function such that $1 \leq t(n) \leq \log_\epsilon n$ 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 $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$.*

Proof: Feige [Fei98] proved that a polynomial time $(1-\delta)\ln n$ approximation algorithm for set cover, for any $\delta > 0$, implies $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$. We obtain such an approximation algorithm assuming there is an algorithm A satisfying the properties of the lemma.

Let I be a set cover instance on n elements. Define $f(n) = n \cdot \epsilon^{-t(\lfloor n \rfloor)}$. Define the sequence of real numbers $n_0 = n$ and $n_{i+1} = f(n_i)$. We solve the instance as follows.

> There will be several *epochs*. Beginning with epoch zero, suppose that at the start of epoch i there are $x_i \leq \lfloor n_i \rfloor$ elements that are uncovered. Take the residual instance (i.e. the remaining elements and sets), add $\lfloor n_i - x_i \rfloor$ new elements, and enlarge all the remaining sets to cover all the new elements (in addition to whatever they covered before). Run A on this instance until at most $\lfloor n_{i+1} \rfloor$ elements remain, then begin the next epoch.

Note that the number of sets in an optimal solution does not increase as the algorithm progresses. Thus, in epoch i at most $t(\lfloor n_i \rfloor)\#OPT(I)$ sets are chosen. Let T be the smallest integer for which $n_T < 1$. It is easy to see that T must exist. Note that $n_T \geq 1/n$ since $t(n) \leq \log_\epsilon n$ and $n_{T-1} \geq 1$. Also, the definition of n_i implies $\sum_{i=0}^{T} t(\lfloor n_i \rfloor) \leq \log_\epsilon(n_0/n_T) \leq 2\log_\epsilon n$. Since we have used at most $\#OPT(I) \cdot \sum_{i=0}^{T} t(\lfloor n_i \rfloor)$ sets, and $\epsilon > e^2$ we obtain a $(1-\delta)\ln n$ approximation for some $\delta > 0$, which concludes the proof. ■

Lemma 2.5 *Suppose $\delta > 0$, and $p(n) = \omega(1)$ is non-decreasing and $1 \leq p(n) \leq (\frac{1}{2} - \delta)\ln n$ 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) be given, fix $\epsilon > e^2$, 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 2.4 and our complexity assumption, we know that for all efficient algorithm A, there exists $n > 0$ such that there exists 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 $\underline{\mathsf{opt}}^p = \sum_{i=1}^{k} i^p |X_i^*| \leq n \cdot [\#OPT(I)]^p$, because the classical solution is also a solution of the L_p version. On the other hand, $\underline{\mathsf{alg}}^p \geq s^p \cdot |R_s|$ for all $s > 0$. In particular, with $s = p \cdot \#OPT(I)$ and our lower bound on $|R_s|$ from Lemma 2.4, we conclude

$$\underline{\mathsf{alg}}^p \geq (\#OPT(I) \cdot p)^p \cdot \frac{n}{\epsilon^p}$$

Therefore, $\underline{\mathsf{alg}}/\underline{\mathsf{opt}} \geq ((p/\epsilon)^p)^{\frac{1}{p}} = p/\epsilon = \Omega(p)$. ■

Lemma 2.6 *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 $\mathbf{NP}$-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 $1 \leq x \leq c_0 t$, 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 that leaves 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 $\mathbf{NP}$-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 $\underline{\mathsf{opt}}^p \leq \sum_{x=1}^{t} \binom{n}{t} x^p$, 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:

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

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 $\underline{\mathsf{alg}}^p/\underline{\mathsf{opt}}^p = \Omega(\left(\frac{p}{e}\right)^p)$, and we obtain a gap of $\underline{\mathsf{alg}}/\underline{\mathsf{opt}} = \Omega(p)$ for all constant p. ■

Combining Lemma 2.5 and Lemma 2.6 immediately yields the following theorem.

Theorem 2.7 *Unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{O(\log\log n)})$, 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 $\Omega(p)$.*

Remark. Note that our $\Omega(p)$ hardness result does not apply when $p = \omega(\log n)$ because the $\log(n)$-norm provides a constant factor approximation for the q-norm for all $q > \log(n)$. Thus, *any* α-approximation algorithm for the $p = \frac{1}{2}\log n$ case outputs solutions which are simultaneously $O(\alpha)$-approximations for all norms in $\{L_p \mid \frac{1}{2}\log n \leq p \leq \infty\} \cup \{L_\infty\}$.

2.3 Extending to L_p Submodular Set Cover

We now consider a generalization of the L_p set cover problem. Our setting now assumes a (monotone) submodular function $f : 2^V \to \mathbb{R}_+$, where $m = |V|$. Recall that f is a monotone submodular function if and only if (1) $f(S) \leq f(T)$ for all $S \subseteq T \subseteq V$, and (2) $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$ for all $S, T \subseteq V$. For a submodular function f, we define a discrete differential as $f_X(Y) = f(X \cup Y) - f(X)$.

The cost of each $v \in V$ is given by the function $c\colon V \to \mathbb{R}_+$. An algorithm picks a permutation $x_1, x_2, \ldots, x_m$ of elements of V. We let $\hat{x}_i = \{x_j : j = 1, \ldots, i\}$. Analogously to the definitions of $|X_i|$ and $|R_i|$ in previous sections, we define $\widetilde{X}_i = f(\hat{x}_i) - f(\hat{x}_{i-1})$ and $\widetilde{R}_i = f(V) - f(\hat{x}_{i-1})$.

For a fixed p, the L_p submodular set-cover problem seeks an arrangement $x_1, \ldots, x_m$ that minimizes the following cost function:

$$\mathsf{cost} = \left(\sum_{i=1}^m s_i^p \widetilde{X}_i\right)^{1/p}, \tag{2.3}$$

where $s_i = \sum_{j=1}^i c_i$. Equivalently, one can express the cost (2.3) in terms of $\widetilde{R}_i$ as $\mathsf{cost} = (\sum_{i=1}^m (s_i^p - s_{i-1}^p)\widetilde{R}_i)^{1/p}$. One can again use the greedy algorithm: at each step choosing $v \in V$ to maximize $f_{\hat{x}_{i-1}}(v)/c(v)$. It is well-known that this algorithm yields an $O(\log n)$ approximation for the L_∞ version [NW81].

Let $\underline{\mathsf{greedy}}$ denote the cost of the greedy algorithm. The quantities $\widetilde{X}_i$, $\widetilde{R}_i$, s_i, x_i, $\hat{x}_i$ correspond those of the greedy algorithm. Similarly, let $\underline{\mathsf{opt}}$ denote the cost of the optimal algorithm, and let $\widetilde{X}_i^*$, $\widetilde{R}_i^*$, s_i^*, x_i^*, $\hat{x}_i^*$ be the corresponding quantities for the optimal algorithm. The proof for the submodular case is similar in spirit to the proof for the L_p SC case. The same analysis in Lemma 2.1 gives the following lemma:

Lemma 2.8 (Bound on Submodular-Greedy Cost) $\underline{\mathsf{greedy}}^p \leq (\underline{\mathsf{greedy}'})^p \stackrel{def}{=} \sum_{i=1}^m \left(p \cdot c(x_i)\frac{\widetilde{R}_i}{\widetilde{X}_i}\right)^p \cdot \widetilde{X}_i$.

The proof is supplied in the appendix for completeness. In order to use the argument of Theorem 2.2, in the following lemma we bound the performance of the optimal algorithm relative to the greedy algorithm.

Lemma 2.9 *Let $i \in [m]$. Let t be the number of elements the optimal algorithm picks before the cost (s_t^*) exceeds $\frac{1}{1+1/p} \cdot \frac{\widetilde{R}_i}{\widetilde{X}_i/c(x_i)}$. Then, $f_{\hat{x}_i}(\hat{x}_t^*) \leq \frac{\widetilde{R}_i}{1+1/p}$*

Proof: Since the greedy algorithm always picks the element that maximizes the ratio between the increase in f and the cost, we know that $f_{\hat{x}_i}(\{x_j^*\})/c(x_j^*) \leq \widetilde{X}_i/c(x_i)$ for all $j = 1, 2, \ldots, t$. It follows that

$$\begin{aligned} f_{\hat{x}_i}(\hat{x}_t^*) &= f(\hat{x}_i \cup \hat{x}_t^*) - f(\hat{x}_i) = \sum_{j=1}^t \left[f(\hat{x}_i \cup \hat{x}_j^*) - f(\hat{x}_i \cup \hat{x}_{j-1}^*)\right] = \sum_{j=1}^t f_{\hat{x}_i \cup \hat{x}_{j-1}^*}(x_j^*) \\ &\leq \sum_{j=1}^t f_{\hat{x}_i}(x_j^*) \leq \frac{\widetilde{X}_i}{c(x_i)} \cdot \frac{1}{1+1/p} \cdot \frac{\widetilde{R}_i}{\widetilde{X}_i/c(x_i)} = \frac{\widetilde{R}_i}{1+1/p}, \end{aligned}$$

which proves the lemma. ■

With all the ingredients we developed so far, we apply the argument of the proof of Theorem 2.2 and obtain the following result, the complete proof of which is given the appendix.

Theorem 2.10 (Submodular L_p Approximation Guarantee) *The greedy algorithm gives a $p \cdot (1+p)^{1/p}(1 + 1/p) \leq 4p$-approximation for the submodular L_p SC 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 can be expressed in terms of the quantities we previously defined as the following:

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

This formulation follows Munagala et al. [MBMW05] but incorporates the notion of cost for each set. [1] Note that when $p = 1$, this cost function is the same that of the L_p case (and the min sum set cover problem). For this problem, we use the technique in the proof of Theorem 2.2 to argue that the greedy algorithm achieves the following approximation ratio; note that no approximation guarantee was given for the general costs case in previous work [MBMW05]. In the interest of space, the proof is given in the appendix.

Theorem 2.11 (Pipelined Set-Cover Approxmation Guarantee) *The (same) greedy algorithm gives a $(1 + 1/p) \cdot (1 + p)^{1/p}$-approximation for the L_p pipelined set-cover problem.*

2.5 A structural lemma for the Greedy Algorithm

We prove a structural lemma for the greedy algorithm, which may be of independent interest. Borrowing notations from previous sections, the lemma assumes all sets have unit costs.

Lemma 2.12 (Set Cover Structure Theorem) *Consider the run of the greedy algorithm at time k, and suppose $|R_k| \in (\frac{n}{2^i}, \frac{n}{2^{i-1}}]$ for some integer i. Fix any increasing function of x, denoted by $F(x)$, and let α be a parameter which is $\Theta(\log F(\log n))$. Then one of the following holds:*

(i) At time $\tau = \frac{k}{\alpha \cdot i}$, there are at least $\frac{F(i) \cdot n}{2^i}$ remaining elements in R^*_τ.
(ii) At time $\tau' = \frac{k}{\alpha}$, there are at least $\frac{n}{2^{i+1}}$ remaining elements in $R^*_{\tau'}$.

We postpone the proof the appendix. In the full version of the paper we show how to use the structural lemma to show that the greedy algorithm is an $O(p \log \log n)$ approximation (a result which is subsumed by Theorem 2.2), and how to show an $O(\log \log n)^2$-AllNorm approximation for Pipelined Set Cover.

3 All L_p Norm Approximations via Sampling Minkowski 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.

Definition 3.1 (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 $X, Y \in \mathbb{R}^n_{\geq 0}$*

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

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

[1] 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.

Theorem 3.2 *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 3.3 (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.*

Due to lack of space, the proofs appear in the appendix. Both the upper bounds are fairly straightforward. However, proving lower bounds is more interesting: if we wanted approximate distance preservation (i.e., for every $X \in \mathbb{R}^n$ and every p, there is a norm $\|\cdot\|_q$ in our sample such that $\|X\|_q \approx \|X\|_p$), it would be easy to see that $\Omega(\log n)$ norms are required. Our lower bound proofs require more work because the notion of α-sampling is weaker than the notion of distance-preservation.

3.1 All L_p Norm Approximations for Facility Location Problems

In this section, we show how the how $O(1)$-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. The following theorem shows how to get an All L_p Norm approximation to such problems.

Theorem 3.4 *There exists a set F of $O(k \log n)$ facilities F such that $\mathsf{Cost}_p(F) \leq O(1) \cdot \mathsf{Cost}_p(\underline{\mathsf{opt}}_p(k))$ 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 3.2 and 3.3, 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 3.5 *Given a parameter α, there exists a metric space over n demand points such that if there is a set of facilities F satisfying*

$$\mathsf{Cost}_p(F) \leq \alpha \cdot \underline{\mathsf{opt}}_p(k), \quad \textit{for all integer } p \geq 1,$$

then

$$|F| \geq \Omega\left(k \left(\frac{\log n}{\log(\alpha k)}\right)^{\frac{1}{3}}\right).$$

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.

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. (One can prove a converse of the above theorem holds true as well; see, e.g., [GM06] for a discussion.)

Theorem 4.1 *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 4.1. An example of a *partial covering* problem is the k-MST problem, where we seek a tree of minimum cost that spans at least k nodes; another example is the k-vertex cover problem, where we seek 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 naturally gives an $O(1)$-submajorization result, and sketch how the same ideas can be extended to other partial cover problems. (The proofs appear in the Appendix C.)

Theorem 4.2 *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 admits an 8-AllNorm approximation, etc. Let us sketch the idea for Vertex Cover: we 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 4.3 *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).*

While we defer the proof to Appendix C.3, let us mention that the techniques we use are fairly natural, and might be of interest to translate other All L_p Norm results to the more general AllNorm setting.

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.

[CGRT03] Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, trees and minimizing latency tours. In *Proceedings of the 44th Symposium on the Foundations of Computer Science (FOCS)*, pages 36–45, 2003.

[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.

[Gar05] Naveen Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 396–402, 2005.

[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 *Algorithms—ESA '06 (Zurich)*, page to appear. Springer, Berlin, 2006.

[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.

[NW81] G. L. Nemhauser and L. A. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. In *Studies on graphs and discrete programming (Brussels, 1979)*, volume 11 of *Ann. Discrete Math.*, pages 279–301. North-Holland, Amsterdam, 1981.

[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.

A Proofs from Section 2

We provide an example for which the greedy algorithm is an $\Omega(p)$ approximation.

Theorem A.1 (Lower Bound for Greedy) *There is a set system on which greedy yields an $\Omega(p)$ approximation.*

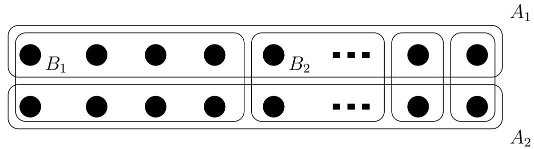

Figure 2: An example that greedy gives an $\Omega(p)$ approximation.

Proof: Consider a set system with $2 \cdot 2^n$ elements. The elements are arranged in two rows, each with 2^n elements. There are two types of sets A and B. As shown in Figure 2, A_1 and A_2 cover the top and the bottom rows, respectively. For $i = 1, 2, \ldots, n+1$, each B_i covers 2^{n-i+1} elements with the exception of B_{n+1} which covers 2 elements. These B_i's are arranged from left to right as shown in the figure.

Let p be given. For sufficiently large n, the optimal algorithm picks A_1 and A_2, which incur a cost of $[2^n(1+2^p)]^{1/p}$. It is easy to see that the greedy algorithm (out of back luck) may pick all the B_i's. Therefore,

$$(\underline{\mathsf{greedy}}/\underline{\mathsf{opt}})^p = \frac{\sum_{i=1}^{n} i^p \cdot 2^{n+1-i} + (n+1)^p \cdot 2^{n+1-n}}{2^n(1+2^p)} \geq \frac{2}{1+2^p} \sum_{i=1}^{n+1} \frac{i^p}{2^i}$$
$$\geq \frac{1}{2^p} \frac{p^p}{2^p} = p^p \cdot 2^{-p-1}.$$

We conclude that $\underline{\mathsf{greedy}}/\underline{\mathsf{opt}} = \Omega(p)$. ∎

A.1 Submodular Set-Cover

We prove Lemma 2.8 here for completeness:

Lemma A.2 (Upper-bound on Submodular-Greedy Cost)

$$\underline{\mathsf{greedy}}^p \;\leq\; (\underline{\mathsf{greedy}}')^p \;\stackrel{def}{=}\; \sum_{i=1}^{m} \left(p \cdot c(x_i) \frac{\widetilde{R}_i}{\widetilde{X}_i} \right)^p \;\cdot\; \widetilde{X}_i.$$

Proof: Let $A_i = \left(p \cdot c(x_i) \frac{\widetilde{R}_i}{\widetilde{X}_i} \right)^p \cdot \widetilde{X}_i$ be the i^{th} term in the summation above. Taking the i^{th} terms of the summation in the definition of greedy's cost, and raising them to the p^{th} powers, define $B_i = (s_i^p - s_{i-1}^p)\, \widetilde{R}_i$ and $C_i = s_i^p\, \widetilde{X}_i$. It follows from Fact 1.1 that

$$\frac{1}{p} A_i + \frac{p-1}{p} C_i \geq A_i^{1/p}\, C_i^{(p-1)/p} = p \cdot c(x_i) \cdot s_i^{p-1} \widetilde{R}_i \geq B_i.$$

The last inequality follows from Fact 1.2 and the observation that $c(x_i) = s_i - s_{i-1}$. Now, rearranging terms, we get that $A_i \geq p\, B_i - (p-1)\, C_i$; summing this over all i and noting that $\sum_i B_i = \sum_i C_i = \underline{\mathsf{greedy}}^p$, we get that

$$\sum_i \left(p \cdot c(\pi(i)) \frac{\widetilde{R}_i^A}{\widetilde{X}_i^A} \right)^p \cdot \widetilde{X}_i^A = \sum_i A_i \geq p \sum_i B_i - (p-1) \sum_i C_i = \underline{\mathsf{greedy}}^p,$$

which completes the proof. ∎

Now we prove Theorem 2.10.

Theorem A.3 (Submodular L_p Approximation Guarantee) *The greedy algorithm gives a $p \cdot (1+p)^{1/p}(1+1/p) \leq 4p$-approximation for the submodular L_p SC problem.*

Proof: The proof is similar in spirit to the proof of Theorem 2.2. We recall that $\underline{\mathsf{greedy}}$ and $\underline{\mathsf{opt}}$ denote the costs of greedy and the optimal algorithm, respectively. The diagram below, which we call the $\underline{\mathsf{opt}}^p$ curve, graphically represents the cost of the optimal algorithm.

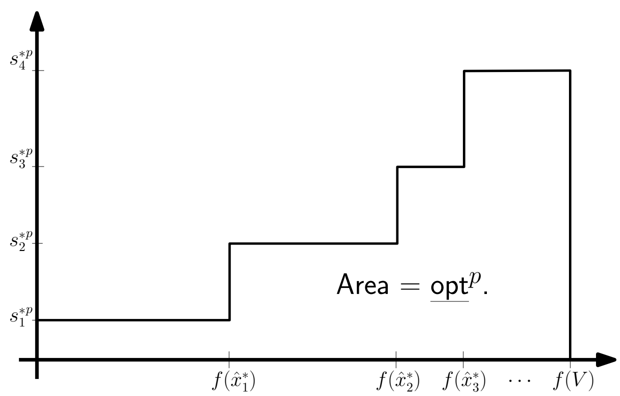

Figure 3: The $\underline{\mathsf{opt}}^p$ curve representing the cost of the optimal algorithm.

We also model the cost of the greedy solution graphically. The figure below has m (unequal) columns corresponding to each element of the universe V. The i-th group has height $[p \cdot c(x_i)\widetilde{R}_i/\widetilde{X}_i]^p$. The area under this curve is $(\underline{\mathsf{greedy}}')^p$.

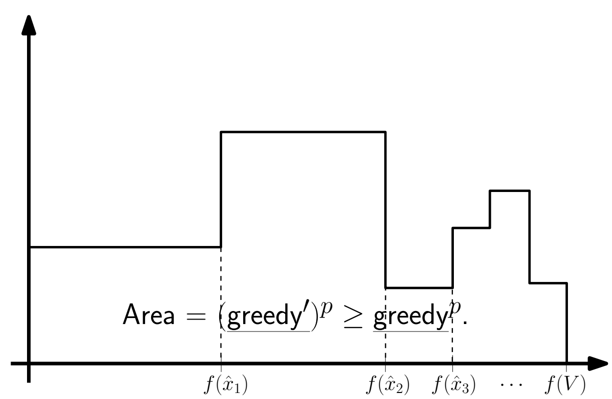

Figure 4: The $\underline{\mathsf{greedy}}'^p$ curve, which upper bounds the cost of the greedy solution.

We will now show that the area of the greedy curve is at most $p^p(1+p)(1+1/p)^p$ times the area of the optimal curve. To prove this, we scale the greedy curve down by $[p(1+1/p)]^p$ vertically and by $1+p$ horizontally. Then, we place this scaled curve so that its bottom-right is aligned with the bottom-right of the $\underline{\mathsf{opt}}^p$ curve. Consider a point $q = (x, y)$ on the original $\underline{\mathsf{greedy}}'^p$ curve. Suppose the point q corresponds to the element x_i, so $y \leq [p \cdot c(x_i)\widetilde{R}_i/\widetilde{X}_i]^p$. Also the distance to q from the right side is at most $\widetilde{R}_i$. Therefore, the height of the point

q after scaling, which we denote by h, is at most

$$\left(\frac{1}{1+1/p} \cdot \frac{\widetilde{R}_i}{\widetilde{X}_i/c(x_i)}\right)^p,$$

and the distance from the right (after scaling), denoted by r, is at most $\widetilde{R}_i/(1+p)$.

To show that a point q of the scaled price curve lies within the $\underline{\mathsf{opt}}^p$ curve, we use the result of Lemma 2.9 and note that

$$\begin{aligned}
\widetilde{R}_t^* &= f(V) - f(\hat{x}_t^*) = f_{\hat{x}_i}(V) - [f(\hat{x}_t^*) - f(\hat{x}_i)] \\
&\geq f_{\hat{x}_i}(V) - [f(\hat{x}_t^* \cup \hat{x}_i) - f(\hat{x}_i)] \\
&= f_{\hat{x}_i}(V) - f_{\hat{x}_i}(\hat{x}_t^*) \\
&\geq \widetilde{R}_i - \frac{1}{1+1/p}\widetilde{R}_i = \frac{\widetilde{R}_i}{1+p}.
\end{aligned}$$

Since $\widetilde{R}_i/(1+p) \geq r$, this implies that q (after scaling) lies within the $\underline{\mathsf{opt}}^p$ curve, and hence the scaled-down version of the greedy curve is completely contained within the optimal curve. Quantitatively, this implies that $\underline{\mathsf{greedy}}^p \leq (1+p)[p(1+1/p)]^p\ \underline{\mathsf{opt}}^p$, which completes the proof. ■

A.2 Pipelined Set-Cover

Theorem A.4 (Pipelined Set-Cover Approxmation Guarantee) *The (same) greedy algorithm gives a $(1+1/p)\cdot(1+p)^{1/p}$-approximation for the L_p pipelined set-cover problem.*

Proof: The proof is similar to our L_p set cover proof. Consider Figure 5, which we call the histogram. The vertical axis represents the cover time. The horizontal axis consists of n^p columns, exactly $|R_i^*|^p - |R_{i+1}^*|^p$ of which have height s_i. The columns are arranged in an increasing order of height, as shown in the figure. Note also that there are $|R_i^*|$ columns with height at least i. By definition, the area under this curve represents the cost of the optimal algorithm.

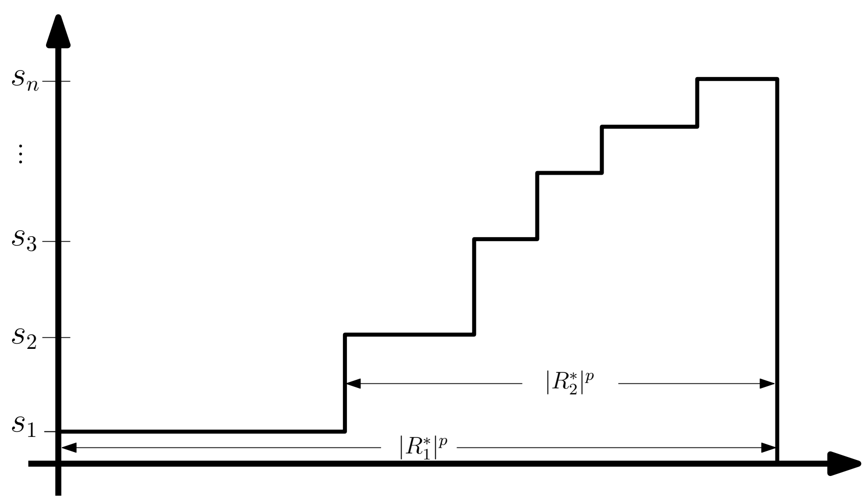

Figure 5: Histogram for the pipelined set cover problem.

We now construct a price curve, which represents the cost of the greedy algorithm. The price diagram has a similar horizontal structure as the histogram: it has n^p columns with exactly $|R_i|^p - |R_{i+1}|^p$ columns in group i. These columns are ordered from left to right in an increasing order of i. The i-th group has height $p_i = \frac{s_i \cdot |R_i|^p}{|R_i|^p - |R_{i+1}|^p}$. It can be easily shown that $p_i \le s_i \cdot |R_i|/|X_i|$ and $\sum p_i$ is the cost of the greedy algorithm.

To finish the proof, we will scale down the price curve and show that it can fit inside the histogram. Let $\gamma = 1/(p+1)$. Scale the price curve down by γ horizontally and $(1-\gamma)^p$ vertically. We will fit the scaled price curve by aligning it to the bottom-right of the histogram. Consider a point p in the i-th part of the price function. If p has height h originally, then $h \le p_i \le s_i|R_i|/|X_i|$. In the scaled version, this corresponds to the time $t = \gamma \cdot h$ in the optimal algorithm. In the scaled version also, the point p is at most $(1-\gamma)^p|R_i|$ from the right.

It is a property of the greedy algorithm that it covers the most fraction of R_i (using X_i); therefore, at time $t = \gamma \cdot h$, the optimal algorithm could have covered—among the elements of R_i—at most $|X_i| \cdot \gamma|R_i|/|X_i| = \gamma|R_i|$ elements. Since the horizontal axis is the p-power of the number of remaining elements, the distance from the right in the histogram at height t is at least $(1-\gamma)^p|R_i|^p$, which shows that p lies within the histogram. Hence, the approximation ratio is $([1-\gamma]^p\gamma)^{-1/p} = \left(1+\frac{1}{p}\right)(1+p)^{1/p}$. ■

A.3 Proof of the Set-Cover Structural Lemma

Lemma A.5 (Set Cover Structure Theorem) *Consider the run of the greedy algorithm at time k, and suppose $|R_k| \in (\frac{n}{2^i}, \frac{n}{2^{i-1}}]$ for some integer i. Fix any increasing function of x, denoted by $F(x)$, and let α be a parameter which is $\Theta(\log F(\log n))$. Then one of the following holds:*

(i) At time $\tau = \frac{k}{\alpha \cdot i}$, there are at least $\frac{F(i)\cdot n}{2^i}$ remaining elements in R^*_τ.
(ii) At time $\tau' = \frac{k}{\alpha}$, there are at least $\frac{n}{2^{i+1}}$ remaining elements in $R^*_{\tau'}$.

Proof: The proof proceeds by assuming that neither of the two events happen, and yield a contradiction by inferring that $|R_k| \le \frac{n}{2^i}$.

First, observe that the elements in $U \setminus R^*_\tau$ are covered by τ sets in the optimal solution. Hence, if we run the greedy algorithm until we pick $\tau' = i \cdot \tau$ sets, Fact C.1 implies that we will have covered at least $(|U \setminus R^*_\tau|) \times (1 - 2^{-i})$ elements from U. Hence the set $R_{\tau'}$ must have at most $|R^*_\tau| + 2^{-i}|U \setminus R^*_\tau|$. By the assumption of the proof, $|R^*_\tau| \le \frac{F(i)\cdot n}{2^i}$. Also, $|U \setminus R^*_\tau| \le |U| = n$, and hence the number of elements in $R_{\tau'}$ satisfies

$$|R_{\tau'}| \le \frac{F(i)\cdot n}{2^i} + \frac{n}{2^i} = \frac{(F(i)+1)\cdot n}{2^i}$$

Similarly, we know that the optimal solution covers the elements of $U - R^*_{\tau'}$, (and hence the elements of $R_{\tau'} - R^*_{\tau'}$) using τ' sets. Hence, applying Fact C.1 $\alpha - 1$ times, we get that by time $\alpha \cdot \tau' = k$, greedy would have at most $|R^*_{\tau'}| + |R_{\tau'} - R^*_{\tau'}|2^{-\alpha+1} \le |R^*_{\tau'}| + |R_{\tau'}|2^{-\alpha+1}$ elements remaining, and thus the number of elements in R_k is at most

$$\frac{n}{2^{i+1}} + \frac{(F(i)+1)\cdot n}{2^i} \times \frac{1}{2^{\alpha-1}} \quad \le \quad \frac{n}{2^i}\left(\frac{1}{2} + \frac{2(F(i)+1)}{2^\alpha}\right). \tag{A.4}$$

Setting $\alpha = \Theta(1) + \log F(\log n)$ would make $2^\alpha \ge 4(F(\log n)+1) \ge 4(F(i)+1)$ and the expression in (A.4) at most $n/2^i$, which would contradict the choice of i, completing the proof. ■

B Proofs from Section 3

B.1 Proofs on Sampling L_p Norms

Theorem B.1 *Given a minimization problem whose objective function is the L_p norm of some cost vector, and an α-sampling S of $D \subseteq \mathbb{R}_{\ge 1} \cup \{\infty\}$, 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 $\{L_p \mid p \in D\}$.*

Proof: Let $\mathbf{C}$ be the cost vector induced by our β-approximation for the class $\{L_p \mid p \in S\}$, and let $\mathbf{C}_p^*$ be the optimal feasible vector for the L_p norm. Fix any $q \in D$ and note that for all $p \in S$, $\|\mathbf{C}\|_p \leq \beta \cdot \|\mathbf{C}_p^*\|_p \leq \beta \cdot \|\mathbf{C}_q^*\|_p$, since $\mathbf{C}_p^*$ is superior to $\mathbf{C}_q^*$ with respect to $\|\cdot\|_p$. Using the α-sampling property with $X = \frac{1}{\beta}\mathbf{C}$ and $Y = \mathbf{C}_q^*$ yields $\|\frac{1}{\beta}\mathbf{C}\|_q \leq \alpha \|\mathbf{C}_q^*\|_q$. Thus $\|\mathbf{C}\|_q \leq \alpha\beta \|\mathbf{C}_q^*\|_q$. ∎

To prove the result on $O(1)$-samplings, let us first give an upper bound on the size.

Theorem B.2 (Upper Bound) *There exists an $O(1)$-sampling S_{reals} of the domain $D_{reals} = \mathbb{R}_{\geq 1} \cup \{\infty\}$ of order n with size $|S| = O(\log n)$. Moreover, there is an $O(1)$-sampling S_{ints} of the domain D_{ints} of order n with size $O(\sqrt{\log n})$.*

Proof: Note that for all p, q such that $1 \leq p \leq q$

$$\|X\|_q \leq \|X\|_p \leq n^{\left(\frac{1}{p}-\frac{1}{q}\right)} \|X\|_q \tag{B.5}$$

and $n^{1/\log n} = O(1)$. With these facts in mind, we define $S_{\text{reals}} := \{\frac{\ln(n)}{k} \mid 1 \leq k \leq \ln(n)\}$ and $S_{\text{ints}} := \{1, 2, \ldots, \sqrt{\ln(n)}\} \cup \{\frac{\ln(n)}{k} \mid 1 \leq k \leq \sqrt{\ln(n)}\}$. We now prove that S_{reals} is an e-sampling of D_{reals}. (We omit the proof for S_{ints}, which is nearly identical). Fix $q \geq 1$, and note that there exists $p \in S_{\text{reals}}$ such that $0 \leq \frac{1}{p} - \frac{1}{q} \leq \frac{1}{\ln(n)}$. Specifically, if $q \geq \ln(n)$, then use $p = \ln(n)$, and if $q \in \left[\frac{\ln(n)}{k}, \frac{\ln(n)}{k-1}\right]$ for some $k \in \{2, 3, \ldots, \ln(n)\}$, then use $p = \frac{\ln(n)}{k}$. Now fix any $X, Y \in \mathbb{R}^n_{\geq 0}$ such that $\|X\|_p \leq \|Y\|_p$. Since $\|X\|_q \leq \|X\|_p$ and $\|Y\|_p \leq n^{\left(\frac{1}{p}-\frac{1}{q}\right)} \|Y\|_q \leq e \cdot \|Y\|_q$ by equation (B.5), we conclude that $\|X\|_q \leq e \cdot \|Y\|_q$. ∎

We now prove lower bounds on the size of $O(1)$-samplings. Suppose we wanted that for every vector X and every value p, there is a norm $\|\cdot\|_q$ in our sample such that $\|X\|_q \approx \|X\|_p$, it is easy to see that $\Omega(\log n)$ norms are required. The lower bound proofs are more complicated because the notion of α-sampling is weaker than the notion of distance-preservation.

Theorem B.3 (Lower Bound) *Every $O(1)$-sampling of $D = \mathbb{R}_{\geq 1} \cup \{\infty\}$ of order n has size $|S| = \Omega(\log n)$. Moreover, every $O(1)$-sampling of $D = \mathbb{Z}_{\geq 1} \cup \{\infty\}$ of order n has size $|S| = \Omega(\sqrt{\log n})$.*

The proof of Theorem B.3 requires the following technical lemma.

Lemma B.4 *For all a, b such that $1 \leq a < b \leq 2a$, there exist $X, Y \in \mathbb{R}^n_{\geq 0}$ such that for all $p \in \mathbb{R}_{\geq 0} \cup \{\infty\} - (a, b)$, $\|X\|_p \leq \|Y\|_p$ and there exists $q \in (a, b)$ such that $\|X\|_q = \Omega(n^{\frac{1}{12} \cdot \frac{b-a}{ab}}) \cdot \|Y\|_q$.*

Proof: Let $D := \mathbb{R}_{\geq 0} \cup \{\infty\}$. We exhibit vectors X, Y such that $\|X\|_p = O(\|Y\|_p)$ for all $p \in D - (a, b)$, and a value $q \in (a, b)$ such that $(\|X\|_q)^q \geq n^{(b-a)/12a} (\|Y\|_q)^q$. Given such a pair of vectors, we can scale X down by $O(1)$ to obtain X', Y such that $\|X'\|_p \leq \|Y\|_p$ for all $p \in D - (a, b)$, and since $q \leq b$, $\|X'\|_q \geq \Omega(n^{(b-a)/12aq}) \|Y\|_q \geq \Omega(n^{(b-a)/12ab}) \|Y\|_q$.

To define the vectors, we will need to define some values. First, fix a, b such that $1 \leq a < b \leq 2a$. Let $b = a + \Delta$.

$$\begin{aligned}
\delta &:= \Delta / a & b &= a + \Delta = a(1+\delta) \\
r &:= \tfrac{1}{a} - \tfrac{\Delta}{4a^2} = \tfrac{1}{a}(1 - \delta/4) & s &:= \tfrac{1}{a} - \tfrac{3\Delta}{4a^2} = \tfrac{1}{a}(1 - 3\delta/4) \\
q &:= a + \tfrac{\Delta}{2} = a(1 + \delta/2) & \tfrac{1}{q} &\approx \tfrac{1}{a} - \tfrac{\Delta}{2a^2}
\end{aligned}$$

The vectors are as follows: $X = (X_1, \ldots, X_n)$ is defined by $X_k = k^{-s}$ for $1 \leq k \leq \tau$, where τ is to be determined later, and $X_k = 0$ otherwise. $Y = (Y_1, \ldots, Y_n)$ is defined by $Y_k = k^{-r}$ for $1 \leq k \leq n$. Note

$$1/b < s < 1/q < r < 1/a$$

First, we set τ to ensure $\|X\|_p = O(\|Y\|_p)$ for all $p \in [1, a]$. Since $ps, pr < 1$

$$\|X\|_p^p = \sum_{k=1}^{\tau} k^{-ps} \approx \frac{\tau^{1-ps}}{1-ps}$$

$$\|Y\|_p^p = \sum_{k=1}^{n} k^{-pr} \approx \frac{n^{1-pr}}{1-pr}$$

Thus if

$$\tau = \min_{p \in [1,a]} \left(\frac{1-ps}{1-pr} \cdot n^{1-pr} \right)^{\frac{1}{1-ps}}$$

then $\|X\|_p = O(\|Y\|_p)$ for all $p \in [1, a]$. Since $s < r$, $\frac{1-ps}{1-pr} \geq 1$. As for the exponent of n, $(1-pr)/(1-ps)$ is minimized in the range $[1, a]$ at $p = a$ and we conclude that

$$\tau \geq n^{(1-ar)/(1-as)} = n^{1/3}$$

(since $1 - ar = \delta/4$ and $1 - as = 3\delta/4$).

Next, we check that $\|X\|_p = O(\|Y\|_p)$ for all $p \in [b, \infty)$. In this case, $ps \geq bs > 1$, and similarly $pr > 1$.

$$\|X\|_p^p = \sum_{k=1}^{\tau} k^{-ps} \approx \frac{1}{ps-1}$$

$$\|Y\|_p^p = \sum_{k=1}^{n} k^{-pr} \approx \frac{1}{pr-1}$$

So we must prove that $\|X\|_p/\|Y\|_p \approx \frac{pr-1}{ps-1} = O(1)$ for all $p \geq b$. It is easy to see that this is the case, since it is upper bounded by $\max\{\frac{br-1}{bs-1}, r/s\}$. Clearly, $r/s < b/a \leq 2$, and $\frac{br-1}{bs-1} \approx \frac{3\delta/4}{\delta/4} = 3$.

Now, we are ready to compute $\|X\|_q^q/\|Y\|_q^q$. Note $s < 1/q < r$, and thus $qs < 1 < qr$

$$\|X\|_q^q = \sum_{k=1}^{\tau} k^{-qs} \approx \frac{\tau^{1-qs}}{1-qs}$$

$$\|Y\|_q^q = \sum_{k=1}^{n} k^{-qr} \approx \frac{1}{qr-1}$$

$$\frac{\|X\|_q^q}{\|Y\|_q^q} \approx \frac{qr-1}{1-qs} \cdot \tau^{1-qs} \approx \frac{\delta/2}{\delta/4} \cdot \tau^{\delta/4} = 2\tau^{\Delta/4a} > n^{\Delta/12a} = n^{(b-a)/12a}$$

Taking the q^{th} root and noting that $n^{(b-a)/12aq} \geq n^{(b-a)/12ab}$ completes the proof. ■

We can now prove the lower bound for *integer* norms.

Theorem B.5 (Lower Bound for Integer Norms) *Every $O(1)$-sampling of $D = \mathbb{Z}_{\geq 1} \cup \{\infty\}$ of order n has size $|S| = \Omega(\sqrt{\log n})$.*

Proof: Suppose S is an α-sampling, and let $\sqrt{\ln n} \leq a < b \leq 2\sqrt{\ln n}$. By Lemma B.4, there exist $X, Y \in \mathbb{R}^n_{\geq 0}$ such that for all $p \in \mathbb{Z}_{\geq 0} \cup \{\infty\} - (a, b)$, $\|X\|_p \leq \|Y\|_p$ and there exists $q \in (a, b)$ such that $\|X\|_q = cn^{\frac{1}{12} \cdot \frac{b-a}{ab}} \cdot \|Y\|_q$ for some constant c. Thus, $cn^{\frac{1}{12} \cdot \frac{b-a}{ab}} \leq \alpha$ implies $(b-a)\ln n \leq 12ab\ln(\alpha/c)$, and thus $b - a \leq 48\ln(\alpha/c)$. It follows that every subinterval of $[\sqrt{\ln n}, 2\sqrt{\ln n}]$ of length greater than $48\ln(\alpha/c)$ must have an element of S in it, and thus $|S \cap [\sqrt{\ln n}, 2\sqrt{\ln n}]| \geq \frac{\sqrt{\ln n}}{48\ln(\alpha/c)} - 1 = \Omega(\sqrt{\ln n})$. ■

Theorem B.6 (Lower Bound for General L_p Norms) *Every $O(1)$-sampling of $D = \mathbb{R}_{\geq 1} \cup \{\infty\}$ of order n has size $|S| = \Omega(\log n)$.*

Proof: Let S be an $O(1)$-sampling of $D = \mathbb{R}_{\geq 1} \cup \{\infty\}$ of order n. We prove that S must contain $\Omega(\log n)$ values in the interval $[1, 2]$. More precisely, we show that S must contain a point in every interval $[a, b]$ with $(b - a) < c/\log n$ for some constant c depending on α, where $1 \leq a < b \leq 2$.

By Lemma B.4, for all $1 \leq a < b \leq 2$, there exist $X, Y \in \mathbb{R}^n_{\geq 0}$ such that for all $p \in \mathbb{R}_{\geq 0} \cup \{\infty\} - (a, b)$, $\|X\|_p \leq \|Y\|_p$ and there exists $q \in (a, b)$ such that $\|X\|_q = \Omega(n^{\frac{1}{12} \cdot \frac{b-a}{ab}}) \cdot \|Y\|_q$. Since $a, b \leq 2$, $\|X\|_q = \Omega(n^{\frac{b-a}{48}}) \cdot \|Y\|_q$. Suppose S is an α-sampling, and $\|X\|_q = c \cdot n^{\frac{b-a}{48}} \cdot \|Y\|_q$. Then $c \cdot n^{\frac{b-a}{48}} \leq \alpha$, and thus $b - a \leq \frac{48 \ln(\alpha/c)}{\ln n}$. It follows that every subinterval of $[1, 2]$ of length greater than $\frac{48 \ln(\alpha/c)}{\ln n}$ must have an element of S in it, and thus $|S \cap [1, 2]| \geq \frac{\ln n}{48 \ln(\alpha/c)} - 1 = \Omega(\ln n)$. ■

B.2 Proofs For Facility Location Problems

Proof of Theorem 3.5: Fix the parameter α. The metric space contains a special point, the *origin* O. The metric space has r *groups* G_i, each group G_i has k *clusters*, and each cluster in G_i has N_i points. (We will refer to a generic cluster in G_i as C_i.) The points within each cluster are at distance zero from each other, points in a cluster C_i are at distance D_i from the origin O, and points in two different clusters C_i and C_j are at distance $D_i + D_j$ from each other. (I.e., distance between any two points in different clusters is the sum of their distances to O).

It remains to set the parameters r, N_i and D_i. It is useful to think of $N_1 \gg N_2 \gg \cdots \gg N_r$ and $D_1 \ll \cdots \ll D_r$). The goal is to show that any solution $\underline{\mathsf{opt}}_p(k + 1)$ (where p is an integer between 1 and r) must open one facility inside each cluster in G_p. (Note the slight change from k to $k + 1$.) Moreover, if we do not open a facility in some cluster of G_p, the cost incurred will be enormous. Hence, if F is a set of facilities satisfying $\mathsf{Cost}_p(F) \leq \alpha \cdot \underline{\mathsf{opt}}_p(k + 1)$ for all values of p, then F must have a facility inside each cluster and so $|F| \geq kr$.

More formally, we wish to set parameters such that for every integer p, $1 \leq p \leq r$, the following inequality holds:

$$N_p \cdot D_p^p \ \geq \ \alpha^p \cdot k \cdot \textstyle\sum_{i:1 \leq i \leq r, i \neq p} N_i D_i^p. \tag{B.6}$$

Indeed, if F is a set of facilities as in the statement of the theorem, then the left hand side of the inequality is a lower bound on $\mathsf{Cost}_p(F)^p$ if there is a cluster C_p in G_p such that F does not contain a facility inside C_p. The right hand side is an upper bound on $(\alpha \cdot \underline{\mathsf{opt}}_p(k + 1))^p$, because the L_p-norm cost of a solution which opens a facility at O and one at each cluster in G_p is at most the p^{th} root of $k \cdot \sum_{i:1 \leq i \leq r, i \neq p} N_i D_i^p$. So if our parameters satisfy (B.6) and F does not open a facility in C_p, we get $\mathsf{Cost}_p(F) \geq \alpha \cdot \underline{\mathsf{opt}}_p(k)$, which violates our assumption about F.

The rest of the proof is just finding the right values for N_i and D_i. Let $\Delta \doteq \alpha \cdot k$, and define $D_i = \Delta^{d_i}$ and $N_i = \Delta^{n_i}$, where $d_i = 2(i - 1)y$, $n_i = x - (i^2 - 1)y$, and x and y are parameters such that d_i and n_i are positive for $i = 1, \ldots, p$. It is routine to check that the inequalities in (B.6) are satisfied if we take $y \geq r$—the key observation is that $N_i D_i^p$ is maximized for $i = p$, and forms a geometric series for $i = 1, \ldots, p - 1$ and for $i = p + 1, \ldots, r$. Finally, we can set $y = r, x = r^3$; the resulting number of nodes is about N_1 which is Δ^{r^3}. Thus, r is $\Omega\left(\frac{\log n}{\log(\alpha k)}\right)^{\frac{1}{3}}$. ■

C Proofs from Section 4

C.1 AllNorm Approximation for Set Cover

We shall show that the greedy algorithm has many nice properties. Let us develop some notation first. Let $\mathcal{O}$ be some algorithm for set cover and $O_1, O_2, \ldots$ denote the sets chosen by $\mathcal{O}$; let $\mathbf{O}$ be the resulting vector of coverage times. Let $\mathcal{A}$ be the greedy algorithm, $A_1, A_2, \ldots$ denote the sets chosen by $\mathcal{A}$, and $\mathbf{A}$ be the coverage times for $\mathcal{A}$. (Hence the element $e_j \in U$ is covered at time $\mathbf{O}_j$ and $\mathbf{A}_j$ in $\mathcal{O}$ and $\mathcal{A}$ respectively.) We shall use the term "time t" (in $\mathcal{O}$ or $\mathcal{A}$) to denote the instant at which exactly t sets have been chosen in the corresponding algorithm. We shall say that an element is *waiting*, *uncovered* or *remaining* at an instant (in $\mathcal{O}$ or $\mathcal{A}$) if it has not been covered yet. We shall use the following easy to prove (and well-known) fact about the greedy algorithm [NW81].

Fact C.1 *Suppose there is a subset $F \subseteq U$ of elements which are all waiting at time t in $\mathcal{A}$. (Note that there might be other elements from $U \setminus F$ waiting at time t as well.) If there are ℓ sets which can cover all the elements in F, then the sets $A_{t+1}, \ldots, A_{t+\ell}$ chosen by the greedy algorithm cover at least $|F|/2$ elements from U (which do not necessarily belong to F).*

One can improve the constant $1/2$ in the above statement, but this will suffice for our analysis. We use the above fact to prove the following simple result.

Theorem C.2 (AllNorm Set Cover) *The AllNorm approximation ratio of the greedy algorithm is $\alpha_{AN}(\mathbf{A}) = O(\log n)$.*

Proof: Consider the solution $\mathbf{A}$ for greedy and the solution $\mathbf{O}$ for any other algorithm $\mathcal{O}$, and $\overleftarrow{\mathbf{A}}$ and $\overleftarrow{\mathbf{O}}$ be these vectors sorted in non-ascending order. If the k^{th} coordinate of $\overleftarrow{\mathbf{O}}$ is t, then at least $n-k+1$ elements of U are covered by t sets. Using Fact C.1, we can infer that the greedy algorithm will cover at least $n-k+1$ elements of U in at most $t\log(n-k+1) \leq t\log n$ sets, and hence the k^{th} coordinate of $\overleftarrow{\mathbf{A}}$ is at most $t\log n$. Using this, we can easily infer that $\mathbf{A}$ is $\log n$-submajorized by $\mathbf{O}$. Finally, using Theorem 4.1 we get that the performance of $\mathcal{A}$ under any symmetric norm is at most $\log n$ times the performance of any other algorithm $\mathcal{O}$. ■

C.2 AllNorm Algorithms from Partial Covering Results

In this section, we will show how solutions for so-called "partial covering" problems can be used to prove submajorization results; by Theorem 4.1, these submajorization results immediately lead to AllNorm approximations for these problems. A *partial covering* problem is one like k-MST where we have to find a good tree that spans at least k nodes, or k-vertex cover where we find a set of nodes that covers at least k edges. In the following, we will show how an $O(1)$-approximation to the k-MST problem naturally gives an $O(1)$-submajorization result; we will then sketch how the same ideas can be extended to other partial cover problems.

AllNorm **TSP.** Consider the TSP problem on a graph $G = (V, E)$. Given a tour π, let t_i be the time at which the salesperson reaches vertex v_i; let $T_\pi = (t_1, t_2, \ldots, t_n)$ be the vector of these *service times* sorted in ascending order.

Theorem C.3 *There is a solution for the TSP problem where the vector of arrival times at each vertex is α-submajorized by the corresponding vector in any other solution, where $\alpha \leq 16$.*

This theorem can be derived from the well-known simple and elegant techniques of Blum et al. [BCC+94], and we give the proof here only for completeness.

Proof: Consider the scheme of Blum et al. [BCC+94], which does the following. For each $i = 0, 1, 2, \ldots$, it wants to find a set of unvisited vertices of maximum size that can be reached with a tour (that returns to the starting point) of length at most 2^i. (The actions done for any fixed value of i are said to occur in *phase-i*.) Since this is an NP-hard problem, we find a tour τ_i of length $c2^i$ (for some suitable constant c) which visits as many previously-unvisited vertices as the best *walk* of length at most 2^i. (Note that we want a *tour* that returns to the origin, whereas we compare ourselves to a *walk* that need not do so.) We then perform the tour τ_i, increment i and iterate until there are no more vertices to visit. For any tour π, we show the above tour α-submajorizes π.

The crucial claim is that if the k^{th} vertex $v_{\pi(k)}$ in π is visited at time in $(2^{i-1}, 2^i]$, then we visit the k^{th} vertex in our algorithm at or before phase-i. Indeed, suppose we have visited only $k' < k$ vertices by phase-$i-1$. The unvisited vertices $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(k)}$ (of which there are at least $k - k'$) can be visited by a *walk* of length at most 2^i, and hence the tour τ_i we find must visit at least $k - k'$ vertices, causing at least k vertices to be visited by the end of phase-i. The total cost incurred by the end of phase-i is $\sum_{j \leq i} c2^i < 2c\,2^i$. Hence the k^{th}-smallest visit time in π is at least 2^{i-1}, whereas the k^{th}-smallest visit time in our tour is at most $2c\,2^i = 4c\,2^{i-1}$, giving us $\alpha = 4c$. Below, we give a subroutine with $c = 4$, hence giving us $\alpha = 4c = 16$.

Finally, it remains to show how find the claimed approximate tour τ_i with $c = 4$: we use an algorithm of Garg [Gar05] for the k-MST problem, which given a number k, finds a tree T_k spanning k vertices, whose cost is within a factor of 2 of optimum. If we run this algorithm for each value of k, take the highest value of k for which the cost of T_k does not exceed 2^{i+1}, and output an Euler tour of this tree T_{k^*}, we will have a tour τ_i of length at most $4 \cdot 2^i$ that visits k^* nodes (i.e., $c = 4$). We claim that the number of nodes visited by any walk of length at most 2^i is at most k^*: indeed, if any walk of length 2^i could visit $k^* + 1$ nodes, this walk would be a valid $(k^* + 1)$-MST solution of cost 2^i, and the 2-approximate $(k^* + 1)$-MST algorithm would have given a solution of cost at most 2^{i+1}, contradicting the choice of k^*. ■

Applying Theorem 4.1, we see that we get a solution vector T_π such that $\alpha_{AN}(T_\pi)$ is a constant. The constant 16 is not the best constant possible (see Chaudhuri et al. [CGRT03] and the references therein); indeed, this proof is given more as a "proof of concept", and we defer the optimizations to the final version.

Extensions to other Partial Covering Problems. To use a similar idea for, say, vertex cover, we 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 use an argument identical to the one above to show 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).

C.3 AllNorm Algorithms for Flow Time on Parallel Machines

In this section, we consider the problem of scheduling jobs over parallel machines. We are given a set of m identical machines. Jobs arrive over time. Let r_i denote the release date of job J_i and p_i its processing requirement. A schedule $\mathcal{A}$ specifies for each machine i and time t, the job which gets processed on this machine at time t. Given a schedule, define the flow-time of a job as the difference between its completion time and release date. With each schedule $\mathcal{A}$, we associate the vector $F^{\mathcal{A}}$ of flow-times of jobs sorted in ascending order. The ℓ_1 norm of this vector is the well-known problem of minimizing the average flow time on parallel machines. Several polynomial-time logarithmic-approximation algorithms for this problem are known (see, e.g., [CKZ01] and the references therein).

However, it is known that for any schedule $\mathcal{A}$, $\alpha_{ALN}(F^{\mathcal{A}})$ is unbounded even if there is only one machine [BP04]. As is standard in such situations, we address this problem by giving our machines $(1+\varepsilon)$-speed where ε is an arbitrary small constant. Bansal and Pruhs [BP04] showed that in the case of single machine, given any constant $\varepsilon > 0$, we can get an $O(\frac{1}{\varepsilon^{O(1)}})$-approximation algorithm for all ℓ_p norms. Chekuri et al. [CGKK04] extended this result to multiple identical machines. We now prove this result for all norms.

For the rest of the discussion, $\mathcal{A}$ shall be used to denote schedules where speed of each machine is $1+\varepsilon$, while $\mathcal{B}$ will denote schedules where speed of each machine is 1. We prove the following theorem.

Theorem C.4 *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).*

Note that such a theorem clearly implies that $\alpha_{AN}(F^{\mathcal{A}})$ is a constant. First, let us mention the basic fact we use to prove this result.

Fact C.5 *A vector A β-submajorizes a vector B if and only if*

$$\sum_i (A_i - z)^+ \leq \sum_i (\beta B_i - z)^+$$

holds for all values of z. Here, $(x)^+$ is defined as $\max\{0, x\}$.

Proof of Theorem C.4: Consider the arguments in the paper of Bansal and Pruhs [BP04], and in the paper of Chekuri et al [CGKK04]. They both define the *age* of a job J_i at time t to be $(t - r_i)$, where r_i is the release time of the job. If p_i is the processing time of the job, then for a fixed parameter α a job is called *mature* if its age is at least $\alpha\, p_i$, and immature otherwise.

The arguments in both papers go as follows: consider any other schedule $\mathcal{B}$. Given any time t, each job J_i that is in $\mathcal{A}$'s queue, but not in $\mathcal{B}$'s queue is either immature, or it can be mapped to a job in $\mathcal{B}$'s queue that has an age of at least $\epsilon'(t - r_i)$. Furthermore, this map is at most $2/\epsilon''$ to 1: i.e., at most $2/\epsilon''$ jobs in the algorithm's queue can be mapped to a job in OPT's queue.

Let us call this map f_t. It will be convenient to view this as a single (partial) map that takes two arguments, a job J_i and a time t, and maps the tuple (J_i, t) to a tuple $(J_{(i,t)^*}, t)$; here the job $J_{(i,t)^*}$ lies in $\mathcal{B}$'s queue also at time t, provided the job J_i is in $\mathcal{A}$'s queue at time t, and is immature. (Furthermore, this map f is at most $2/\epsilon''$-to-1.

Let $F^{\mathcal{A}} = (a_1, a_2, \ldots, a_n)$ be the vector of flow times of vectors according to the algorithm $\mathcal{A}$ sorted in non-increasing order; let the jobs be renumbered so that J_i is the job with flow time a_i. Let B be any vector of flow times (sorted in non-increasing order). Define $F^{\mathcal{B}} = (b_1, \ldots, b_n)$ similarly. We will use Fact C.5 to prove the submajorization result.

Let us fix a value of z, and look at any job that contributes to $\sum_i (a_i - z)^+$. Each time slot (J_i, t) of this contribution is either mature or it is not. Note that the fact that this slot is contributing implies that $(t - r_i) > z$. If it is mature, then $(J_i, t) \mapsto (J_{(i,t)^*}, t)$ such that the age of $J_{(i,t)^*}$ at time t is at least $\epsilon'(t - r_i) > \epsilon'\, z$. And hence the slot $(J_{(i,t)^*}, t)$ would contribute to the sum $\sum_i ((1/\epsilon')b_i - z)^+$. However, since the map is $2/\epsilon''$ to 1, the contribution should be scaled down by a factor of at most $\epsilon''/2$, and thus the contributions of the mature slots to $\sum_i (a_i - z)^+$ is at most

$$(2/\epsilon'') \times \sum_i ((1/\epsilon')b_i - z)^+ \leq \sum_i (((2/(\epsilon' \times \epsilon'')) \times b_i - z)^+ . \tag{C.7}$$

On the other hand, if the slot (J_i, t) contributes to $\sum_i (a_i - z)^+$ but is not mature, then $z < (t - r_i) \leq \alpha p_i$. Hence the contribution of the immature slots to $\sum_i (a_i - z)^+$ is $\sum_i (\alpha\, p_i - z)^+$. However, since the flow time of each job J_i in $\mathcal{B}$ is at least p_i, it follows that the contribution for the immature jobs is bounded by $\sum_i (\alpha\, b_i - z)^+$. Combining these two expressions together, we get that

$$\sum_i (a_i - z)^+ \leq \sum_i (\beta\, b_i - z)^+,$$

where $\beta = (2/(\epsilon'\epsilon'') + \alpha)$. ■