Algorithms for Hub Label Optimization

MAXIM BABENKO, National Research University Higher School of Economics
ANDREW V. GOLDBERG, Amazon.com
ANUPAM GUPTA, Carnegie Mellon University
VISWANATH NAGARAJAN, University of Michigan

We consider the hub label optimization problem, which arises in designing fast preprocessing-based shortest-path algorithms. We give $O(\log n)$-approximation algorithms for the objectives of minimizing the maximum label size (ℓ_∞-norm) and simultaneously minimizing a constant number of ℓ_p-norms. Prior to this, an $O(\log n)$-approximation algorithm was known [Cohen et al. 2003] only for minimizing the total label size (ℓ_1-norm).

CCS Concepts: • **Theory of computation → Shortest paths**; **Approximation algorithms analysis;**

Additional Key Words and Phrases: Labeling algorithms, distance oracles

ACM Reference Format:
Maxim Babenko, Andrew V. Goldberg, Anupam Gupta, and Viswanath Nagarajan. 2016. Algorithms for hub label optimization. ACM Trans. Algorithms 13, 1, Article 16 (November 2016), 17 pages.
DOI: http://dx.doi.org/10.1145/2996593

1. INTRODUCTION

Modern applications, such as computing driving directions and other location-based services, require very fast point-to-point shortest path algorithms. Although Dijkstra's algorithm solves this problem in near-linear time [Goldberg 2008] on directed and in linear time on undirected graphs [Thorup 1999], some applications require sublinear distance queries. This motivates preprocessing-based algorithms, which yield sublinear queries on some graph classes (e.g., Delling et al. [2009] and Gavoille et al. [2004]). In particular, Gavoille et al. [2004] introduced *distance labeling* algorithms. These algorithms precompute *labels* for each vertex such that the distance between any two vertices s and t can be computed using only their labels.

A prominent case of this paradigm is *hub labeling (HL)*: The label of v consists of a collection of vertices (the *hubs* of v) with their distances from v. Hub labels satisfy the *cover property*: For any two vertices s and t, there exists a vertex w on the shortest s–t path that belongs to both the label of s and the label of t. Given this information, distance queries are easy to implement: For two vertices u and v, we compute the sums of the u-w and w-v distances over vertices w in the intersection of the labels of u and v and return the minimum value found.

A. Gupta's research was supported in part by NSF awards CCF-0964474 and CCF-1016799.
Authors' addresses: M. Babenko, National Research University Higher School of Economics (HSE) and Yandex, Moscow 125319, 3 Kochnovskiy Proezd, 607, Russia; email: maxim.babenko@gmail.com; A. V. Goldberg, Amazon.com, 1900 University Ave, E. Palo Alto, CA 04303, USA; email: avg@alum.mit.edu; A. Gupta, Computer Science Department, Carnegie Mellon University, 7203 Gates Building, Pittsburgh PA 15213, USA; email: anupamg@cs.cmu.edu; V. Nagarajan, Industrial and Operations Engineering Department, University of Michigan, 1205 Beal Ave, Ann Arbor, MI 48109, USA; email: viswa@umich.edu.

DOI: http://dx.doi.org/10.1145/2996593

Cohen et al. [2003] gave an $O(\log n)$-approximation algorithm for the smallest-size labeling, where n denotes the number of vertices and the size of the labeling is the sum of the number of hubs in the vertex labels. (This also minimizes the average label size.) The algorithm uses an elegant reduction to the set-cover problem [Chvátal 1979]. At each step, the algorithm solves a maximum density subgraph problem, which can be done exactly using parametric flows [Gallo et al. 1989] or by a faster approximation algorithm [Kortsarz and Peleg 1994]. HL leads to the fastest implementation of the point-to-point shortest path queries in road networks [Abraham et al. 2011] and works well on some other network types [Abraham et al. 2012]. This motivates further theoretical study of HL.

In this article, we consider approximation algorithms for the optimization problem of producing small labels. Since minimizing the average label size may potentially lead to imbalanced solutions where the label of some vertices are relatively large, a natural objective is to minimize the maximum HL size, which determines the worst-case query time. We give a polynomial time algorithm that finds an $O(\log n)$ approximation of the maximum label size, where n is the size of the graph. Our algorithm is based on reducing the HL problem to a non-standard set covering problem (called the *low-load set-covering problem*, or *LSC* for brevity), where the objective is to cover all the elements using sets in such a way that no element is covered too often. Our algorithm is combinatorial (does not rely on general linear or convex programming solvers) and is based on exponential cost functions, as in Aspnes et al. [1997] and many other contexts.

If we consider a vector whose components correspond to the label sizes at vertices, then the total label size is the ℓ_1-norm of this vector, and the maximum label size is the ℓ_∞-norm. This brings a natural generalization of the above problems, that of optimizing the ℓ_p-norm of this vector. Our second result is an $O(\log n)$-approximation algorithm for this more general problem. This is also a combinatorial algorithm, where we use degree-p polynomials instead of exponential cost functions, as in Awerbuch et al. [1995].

In applications, there are multiple criteria that one would like to be good for—one wants to simultaneously minimize the total label size (i.e., the space needed to store the labels) and the maximum label size (i.e., the worst-case query time). This is a bi-criteria optimization problem, with the “optimal solutions” on a Pareto-optimal curve. Our third result is a polynomial-time algorithm that approximates the Pareto curve. For example, suppose there is a labeling (on the Pareto curve) with total label size T_1 and maximum label size T_∞; then our algorithm finds a labeling with total label size $O(T_1 \log n)$ and maximum label size $O(T_\infty \log n)$. In fact, our techniques easily extend to a more general result: a logarithmic approximation to the problem of maintaining k moments of the label sizes. Specifically, given any set $P = \{p_1, p_2, \dots, p_k\}$, and values T_i such that there exists a labeling whose ℓ_{p_i}-norm is at most T_i for each $p_i \in P$, we can find a labeling whose ℓ_{p_i}-norm is at most $O(k \log n) \cdot T_i$ for all $p_i \in P$.

An alternative approach to obtaining polynomial-time logarithmic approximations for some of our labeling problems is via linear programming. One can write a linear program with exponentially many variables, solve it approximately using a separation oracle, and then use randomized rounding. Even using the current theoretically fastest linear programming algorithms [Lee et al. 2015], the running time of such an approach would be several logarithmic factors larger than the algorithms obtained this article. Moreover, our algorithms are much simpler to implement than using the linear-program-based approach.

1.1. Related Work

There is much work on minimizing multiple norms of a vector. For some problems, one can find a single solution that is simultaneously good against the best solution for each ℓ_p-norm individually. For example, Azar et al. [2004] considered the restricted-assignment machine scheduling problem and gave a solution that 2-approximates the best solution for each ℓ_p-norm; this was extended and improved by Azar and Epstein [2005] and Kumar et al. [2009]. In contrast, such strong guarantees are not possible for the LSC problem (even in the special case of HL), as we show in Section 7.1.

In some settings, it is possible to get better bounds for ℓ_p-norm minimization for different values of p: For example, Awerbuch et al. [1995] showed tight p-competitive *online* algorithms for minimizing the ℓ_p-norms of machine loads. However, the LSC problem has an $\Omega(\log n)$-hardness of approximation for all p, which means that new techniques are needed here.

1.2. Outline

We formally define the HL and LSC problems in Section 2. In Section 3, we give the reduction from HL to LSC. Section 4 contains the $O(\log n)$-approximation algorithm for minimizing the maximum load (ℓ_∞-norm) in LSC. Section 5 gives an algorithm for simultaneously minimizing both total (ℓ_1-norm) and maximum (ℓ_∞-norm) load, which yields an $O(\log n)$-approximation to the Pareto curve. Then, Section 6 gives an $O(\log n)$-approximation algorithm for LSC under any ℓ_p-norm objective. This gives an $O(\log n)$-approximation algorithm for HL via the above reduction. In Section 6.1, we outline a faster implementation of this algorithm for HL.

In Section 7, we extend the LSC approximation algorithm to simultaneously minimize any constant number of ℓ_p-norms. Finally, in Section 7.1, we show that there are instances of hub labeling (even on undirected graphs) where no labeling can simultaneously achieve both minimum possible total and maximum load. Observe that our result on constant number of ℓ_p-norms (Theorem 7.2) generalizes all our previous results in this article. However, we present the results incrementally in order to make the article more readable.

2. DEFINITIONS AND NOTATION

In the HL problem, we are given a graph on vertices V with a distinguished shortest path P_{ij} between each pair of vertices $\{\{i, j\} : i, j \in V\}$. We use $n = |V|$ to denote the number of vertices. A hub labeling is an assignment of labels $L_i \subseteq V$ for each vertex $i \in V$ such that for any $i, j \in V$, we have some vertex $u \in L_i \cap L_j$ that lies on the path P_{ij}. For the purposes of this article, the graph can be directed or undirected, and the path P_{ij} can be an arbitrary path—we will not use the fact that it is a shortest i-j path. If we consider the vector $\mathbf{L} = (|L_1|, |L_2|, \ldots, |L_n|)$, then we are interested in finding labelings with small ℓ_p-norm: $\|\mathbf{L}\|_p := (\sum_{i=1}^n |L_i|^p)^{1/p}$ and $\|\mathbf{L}\|_\infty := \max_{i\in V} |L_i|$.

We assume $p \in [1, \log n]$, since $\ell_{\log n}$ approximates all higher ℓ_p-norms to within constant factors. Indeed, for any $x \in \mathbb{R}^n$ and any $p \geq \log n$,

$$\|x\|_p \leq \|x\|_{\log n} \leq \|x\|_\infty \|\mathbf{1}\|_{\log n} = \|x\|_\infty n^{1/\log n} \leq e\|x\|_\infty \leq e\|x\|_p.$$

Above, the first and last inequalities is from the monotonicity of ℓ_p-norms, and the vector $\mathbf{1} \in \mathbb{R}^n$ is the vector of all 1s.

We will reduce HL to the low-load set-covering problem (LSC), which is defined as follows. As in the usual set cover problem, we are given a set system $(U, \mathcal{F})$, where U denotes a universe of elements and $\mathcal{F} = \{S_1, S_2, \ldots\}$ is a collection of subsets of this universe. We use $N = |U|$ to denote the size of the universe. A sub-collection $\mathcal{C} \subseteq \mathcal{F}$ is a *set cover* if $\cup_{S\in\mathcal{C}} S = U$, that is, every element of U is contained in some set of $\mathcal{C}$. Furthermore, U is partitioned into *relevant* elements (denoted $R \subseteq U$) and *irrelevant* elements (those in $U \backslash R$). For any set cover $\mathcal{C}$ and relevant element $e \in R$, let $A_\mathcal{C}(e) = |\{S \in \mathcal{C} \mid e \in S\}|$ be the *load* of element e under $\mathcal{C}$; this is the number of sets in $\mathcal{C}$ that contain e. (We do not care about the number of times irrelevant elements are covered, so one can imagine their load as always zero.) For any $p \in [1, \infty)$, the

ℓ_p-norm of the loads is $\|A_{\mathcal{C}}\|_p = (\sum_{e \in R} A_{\mathcal{C}}(e)^p)^{1/p}$; the ℓ_∞-norm is $\|A_{\mathcal{C}}\|_\infty = \max_{e \in R} A_{\mathcal{C}}(e)$. We will be interested in finding a set cover $\mathcal{C}$ having minimum ℓ_p-norm $\|A_{\mathcal{C}}\|_p$ for $p \in [1, \log N] \cup \{\infty\}$; again, the $\ell_{\log N}$-norm approximates ℓ_∞ to within a constant factor. To reiterate: We have to cover all the elements, relevant or otherwise, but the objective involves only the loads on relevant elements.

As in many set-cover based algorithms, we make use of min-ratio oracles. The *min-ratio problem* takes as input an instance of the LSC problem, element costs $c_e \in \mathbb{R}_+$ for the relevant elements $e \in R$, and a set $X \subseteq U$ of elements already covered and seeks to output a set $S \in \mathcal{F}$ that minimizes the cost-to-coverage ratio,

$$\frac{\sum_{e \in S \cap R} c_e}{|S \setminus X|}.$$

A *min-ratio oracle* is simply an algorithm that solves the min-ratio problem. In Section 3.2, we show how to implement a min-ratio oracle for the LSC instances that arise out of reductions from HL.

3. APPLICATION TO SHORTEST PATH LABELS

Our motivating application for the low-load set cover problem is HL ℓ_p-norm optimization. We show a reduction from the label optimization problem to LSC and an implementation of a min-ratio oracle for the corresponding LSC problem.

3.1. From Labels to Set Covers

We model an instance $\mathcal{I}$ of HL as the following instance $\mathcal{I}'$ of LSC.

—The elements are all $\{i, j\}$ pairs and all vertices; so the universe $U = \binom{V}{2} \cup V$. The elements of V are relevant and the elements of $\binom{V}{2}$ are irrelevant.
—For each vertex $x \in V$, let Q_x be the set of pairs $\{i, j\} \in \binom{V}{2}$ such that $x \in P_{ij}$. For any set of pairs $Q \subseteq \binom{V}{2}$, let $V(Q)$ denote the vertices that lie in at least one of these pairs—that is, $V(Q) = \cup_{\{i,j\} \in Q} \{i, j\}$. Now for each $x \in V$, for each $Q \subseteq Q_x$, the set $Q \cup V(Q)$ is in the collection $\mathcal{F}$. Note that this gives us exponentially many sets.
—The goal is to compute a set cover $\mathcal{C} \subseteq \mathcal{F}$ minimizing the ℓ_p-norm $\|A_{\mathcal{C}}\|_p$ of element-loads (recall that we only consider the loads on relevant elements, i.e., vertices).

LEMMA 3.1. *Minimizing the ℓ_p-norm of an instance $\mathcal{I}$ of HL is equivalent to minimizing the ℓ_p-norm of the corresponding instance $\mathcal{I}'$ of LSC.*

PROOF. Given a solution $\{L_i \subseteq V\}_{i \in V}$ for the HL instance $\mathcal{I}$, construct a solution $\mathcal{C}$ for the LSC instance $\mathcal{I}'$ as follows. For each $x \in V$, take $S_x \subseteq \binom{V}{2}$ to be the pairs that x "covers" in this solution—that is, pairs $\{i, j\}$ such that x lies in $L_i \cap L_j$ and also on the path P_{ij}. Solution $\mathcal{C}$ consists of the sets $S_x \cup V(S_x)$, for each $x \in V$. Note that $\mathcal{C}$ is a valid set-cover for $\mathcal{I}'$. If the label of i does not contain x, then $i \notin V(S_x)$. Thus only the vertices $x \in L_i$ can contribute to $A_{\mathcal{C}}(i)$, and so $A_{\mathcal{C}}(i) \leq |L_i|$. Hence $\|A_{\mathcal{C}}\|_p \leq \|\mathbf{L}\|_p$.

Given a solution $\mathcal{C}$ to $\mathcal{I}'$, we construct a labeling $\{L_i \subseteq V\}_{i \in V}$ for $\mathcal{I}$ as follows. If $\mathcal{I}'$ contains a set $(Q \cup V(Q))$ for some $Q \subseteq Q_x$ and $x \in V$, then add x to L_i for all $i \in V(Q)$. Since all pairs in U are covered by $\mathcal{C}$, for each pair $\{i, j\}$, there is some $(P, V(P)) \in \mathcal{C}$ where $P \subseteq Q_y$, $y \in V$ and $\{i, j\} \in P$. In other words, for each pair $\{i, j\}$ there is some $y \in P_{ij} \cap L_i \cap L_j$. So this is a valid labeling for the HL instance $\mathcal{I}$. For every vertex i, note that we add $x \in V$ to L_i only if there is a set $(Q \cup V(Q)) \in \mathcal{C}$ with $Q \subseteq Q_x$ and $i \in V(Q)$; so $|L_i| \leq A_{\mathcal{C}}(i)$. Hence $\|\mathbf{L}\|_p \leq \|A_{\mathcal{C}}\|_p$. $\square$

3.2. Min-Ratio Oracle

In this section, we show how to construct approximate min-ratio oracles for the LSC instances obtained via the above reduction from HL.

Recall that the universe $U = \binom{V}{2} \cup V$, where V is the vertex-set of the HL instance. For any set $S \subseteq U$, we use $S_P := S \cap \binom{V}{2}$ to denote the pairs in S (i.e., the irrelevant elements) and $S_V := S \cap V$ to denote the vertices in S (i.e., the relevant elements). The min-ratio oracle takes input costs $c_u > 0$ for all $u \in V$ (relevant elements), and a subset $X \subseteq U$. The goal is to find a set $S \in \mathcal{F}$ that minimizes the ratio $\frac{\sum_{u \in S_V} c_u}{|S \setminus X|}$. We next show that this can be solved exactly in polynomial time. This is similar to the subroutine used in Cohen et al. [2003] for minimizing the ℓ_1-norm of labels. (In their setting vertex costs were always unit.)

For any subgraph $H \subseteq G$ its density is defined to be $\mu(H) = \frac{|E(H)|}{c(V(H))}$. The min-ratio algorithm relies on an algorithm for the *weighted maximum density subgraph* (WMDS) problem: Given an undirected graph $G = (V, E)$ with a non-negative cost function $c : V \to \mathcal{R}_+$, the goal is to find a vertex-induced subgraph H that maximizes $\mu(H)$. We note that the graph G may contain self-loops. The WMDS problem is known to be solvable exactly using network flows [Gallo et al. 1989]. In Section 3.3, we describe a faster 2-approximation algorithm that generalizes a known algorithm [Kortsarz and Peleg 1994] for unit-cost WMDS.

Given a set $X \subseteq U$ of covered elements as input to the min-ratio oracle, for any vertex $v \in V$ define the *v-center* graph G_v as follows. The vertex set of G_v is V. For each pair of vertices $i, j \in V$ with $P_{i,j} \ni v$ (i.e., v covers the pair) G_v contains the following edges:

—an edge (i, j) iff $\{i, j\} \in \binom{V}{2} \setminus X_P$, and
—a self-loop (i, i) (respectively, (j, j)) iff $i \in V \setminus X_V$ (respectively, $j \in V \setminus X_V$).

THEOREM 3.2. *If there is a ρ-approximation algorithm for the weighted maximum density subgraph problem, then there is a ρ-approximation algorithm for the min-ratio oracle* $\min_{S \in \mathcal{F}} \frac{\sum_{u \in S_V} c_u}{|S \setminus X|}$ *for HL with arbitrary positive weights c.*

PROOF. By definition of the set collection $\mathcal{F}$, it can be partitioned into $\{\mathcal{F}_v : v \in V\}$, where

$$\mathcal{F}_v = \{Q \cup V(Q) \, : \, Q \subseteq Q_v\}.$$

Recall that Q_v is the set of all pairs $\{i, j\} \in U_P$ such that $v \in P_{ij}$, and $V(Q)$ is the set of vertices that appear in some pair of Q. We will show that for any $v \in V$, solving $\min_{S \in \mathcal{F}_v}(\sum_{u \in S_V} c_u \,/\, |S \setminus X|)$ corresponds to an instance of WMDS. The min-ratio algorithm solves each of the n WMDS instances using the ρ-approximation algorithm and returns the best solution found.

Fix any $v \in V$. Observe that any set $S \in \mathcal{F}_v$ has $S_P = Q$ for some $Q \subseteq Q_v$ and $S_V = V(Q)$; so $S \setminus X = (S_P \setminus X_P) \bigcup (S_V \setminus X_V)$. By definition of graph G_v, it follows that any set $S \in \mathcal{F}_v$ corresponds to the subgraph H_S on vertices S_V and edges $(S_P \setminus X_P) \bigcup (S_V \setminus X_V)$. Thus $\frac{\sum_{u \in S_V} c_u}{|S \setminus X|} = 1/\mu(H_S)$. Conversely, any subgraph of G_v corresponds to a set in $\mathcal{F}_v$. Hence the min-ratio problem restricted to sets in $\mathcal{F}_v$ is exactly the WMDS problem on G_v. □

Note that the results of this section extend to directed graphs; in this case, the center graphs will become bipartite, as in Cohen et al. [2003].

3.3. Faster Approximate Algorithm for Weighted Maximum Density Subgraph

The 2-approximation algorithm for WMDS works iteratively as follows. The algorithm maintains a current graph $G' = (V', E')$, which is initially $G' = G$. Let $\delta'(v) = |\{(u, v) \in E' : u \in V'\}|$ denote the degree of v in G' (this includes any self-loop). In each iteration, the algorithm finds a vertex v in G' that minimizes $\delta'(v)/c(v)$. The algorithm deletes

v from G' to obtain the new current graph G'. The algorithm also remembers G'', the maximum density subgraph G' seen so far. When G' contains a single vertex, the algorithm halts and returns G''.

We start the proof of correctness with the following lemma.

LEMMA 3.3. *Let $G^* = (V^*, E^*)$ be a WMDS of G' and let v satisfy $\frac{\delta'(v)}{c(v)} < \frac{|E^*|}{c(V^*)}$. Then $v \notin V^*$.*

PROOF. Suppose for contradiction that $v \in V^*$. Since G^* is a subgraph of G', we have $\delta'(v) \geq \delta^*(v)$ the degree of v in G^*. So

$$\frac{\delta^*(v)}{c(v)} < \frac{|E^*|}{c(V^*)}.$$

We can rewrite this as $c(V^*)\delta^*(v) < c(v)|E^*|$. Subtracting both sides from $c(V^*)|E^*|$, we get $c(V^*)|E^*| - c(V^*)\delta^*(v) > c(V^*)|E^*| - c(v)|E^*|$. This implies

$$\frac{|E^*| - \delta^*(v)}{c(V^*) - c(v)} > \frac{|E^*|}{c(V^*)}.$$

The left-hand side is the density of G^* with v deleted, and the right-hand side is the density of G^*. This contradicts the optimality of G^*. □

LEMMA 3.4. *The above algorithm achieves a 2-approximation for WMDS.*

PROOF. If the optimal density is ∞, then the algorithm finds the optimal subgraph because it removes zero-weight vertices with positive degree last. If the graph G has no edges, then also the algorithm finds the optimal subgraph. Now assume that the optimal density is positive and finite.

Let G^* denote the maximum density subgraph of G and let μ^* be the density of G^*. Consider the first time that the algorithm removes a vertex from G^*; note that G^* is also the maximum density subgraph of the current graph G' in this iteration. By Lemma 3.3 and the rule for removing vertices, we have $\delta'(v) \geq \mu^* c(v)$ for any $v \in V'$. So the density of graph G' in this iteration is

$$\frac{|E'|}{c(V')} \geq \frac{\sum_{v \in V'} \delta'(v)}{2c(V')} \geq \frac{\mu^* \sum_{v \in V'} c(v)}{2c(V')} = \frac{\mu^*}{2}.$$

The result follows since the algorithm outputs the best ratio subgraph among all the G's. □

The original 2-approximation algorithm for the unit-weight maximum density subgraph problem runs in linear time because, using the bucket data structure, the work of finding the minimum degree vertex v can be charged to the deletion of edges adjacent to v. The weighed version of the algorithm has a slightly higher complexity due to the data structure overhead.

THEOREM 3.5. *There is a 2-approximation algorithm for weighted maximum density subgraph that runs in $O(m + n \log n)$ time.*

PROOF. As described, the algorithm needs to maintain G'', the best subgraph seen so far, which may be expensive. However, we can avoid maintaining G'' by reconstructing it at the end of the algorithm. We run the above algorithm and compute a vertex ordering σ. Then, we consider the vertices in the order σ and compute the density of each suffix in this ordering (by removing each vertex after it is considered). Finally, we output the graph G'' corresponding to the suffix which has the maximum density.

The only non-trivial part in analyzing the running time is in computing the vertex ordering σ. (It is easy to see that the other steps only take $O(m + n)$ time.) We now describe the implementation details in computing σ. We maintain a priority queue with amortized operation costs of $O(1)$ for the insert and decrease-key operations and $O(\log n)$ for the extract-min operation (e.g., a Fibonacci heap [Fredman and Tarjan 1987]). The queue contains vertices with keys $\delta'(v)/c(v)$. Each vertex is inserted into the queue once and extracted at most once; the total cost of these operations is $O(n \log n)$. Decrease-key operations happen when the algorithm deletes a neighbor w of a vertex v and can be charged to the edge (v, w) for a total of $O(m)$ work. This yields the $O(m + n \log n)$ bound. □

Combining the bound of the theorem with Theorem 3.2 and the observation that the graph we apply the algorthm to has $O(n^2)$ arcs, we get the following result:

COROLLARY 3.6. *There is an $O(n^3)$ implementation of the minimum ratio oracle for HL.*

4. THE ℓ_∞ CASE: MINIMIZING THE MAXIMUM LOAD

In this section, we investigate the problem of finding an LSC solution $\mathcal{C}$ that approximately minimizes the maximum load of any relevant element in U. Recall that we have to cover the irrelevant elements (those in $U \setminus R$), even though we do not care about the load on them. Suppose that we know the optimal load $u = \|A_{\mathcal{C}^*}\|_\infty$; we can enumerate over all the possible values of u. We show a combinatorial greedylike algorithm that achieves an approximation ratio of $\rho = O(\log N)$, where $N = |U|$.

This result is asymptotically best possible, since (as shown next) the LSC problem is at least as hard to approximate as the usual set cover problem. Consider any instance $(U, \mathcal{F})$ of the usual set-cover problem, with universe U and collection of subsets $\mathcal{F}$; the goal is to find a minimum cardinality sub-collection of $\mathcal{F}$ that contains every element of U. We now define an instance of LSC on universe $U' = U \cup \{e_0\}$ for a new element e_0. There is a set $S \cup \{e_0\}$ for each $S \in \mathcal{F}$, and e_0 is the only relevant element. Notice that any set-cover $\mathcal{C} \subseteq \mathcal{F}$ corresponds to a feasible LSC solution with ℓ_p-norm (for any p) equal to $|\mathcal{C}|$. Thus LSC under any ℓ_p-norm is $\Omega(\log N)$ hard to approximate [Feige 1998].

Since the family $\mathcal{F}$ in an LSC instance may consist of an exponential number of sets, it may not be possible to explicitly look over all sets. We assume that we have an α-approximate *min-ratio oracle* through which we access the set system.

THEOREM 4.1. *There is an $O(\alpha \log N)$-approximation algorithm for LSC under ℓ_∞-norm that makes N calls to an α-approximate min-ratio oracle.*

The algorithm uses multiplicative-weight updates. It proceeds in *rounds*, where one set is added in each round. Let $\varepsilon = 1/(8\alpha)$. Let $A_t(e)$ be the number of times a relevant element $e \in R$ has been covered at the beginning of round t; at the very beginning of the process we have $A_1(e) = 0$ for all $e \in R$. Define the round-t cost of elements $e \in R$ as $c_t(e) := (1+\varepsilon)^{A_t(e)/u} \cdot ((1+\varepsilon)^{1/u} - 1)$; so the round-$t$ cost of set S is

$$c_t(S) := \sum_{e \in S \cap R} c_t(e) = \sum_{e \in S \cap R} (1+\varepsilon)^{A_t(e)/u} \cdot \big((1+\varepsilon)^{1/u} - 1\big).$$

Note the sum is only over the relevant elements in S. This choice of the cost function is based on the well-known "soft max" function $\ln \sum_e \exp(x_e) \approx \max_e\{x_e\}$; we use the constant $1 + \varepsilon$ as the base of the exponent because it simplifies the analysis. To get more intuition for this expression, observe that if the coverage of a relevant element increases *additively* by u, its cost increases *multiplicatively* by $(1 + \varepsilon)$. This rapid increase disincentivizes our algorithm from picking sets that cover this element too many times.

The algorithm is a simple greedy one: Consider the beginning of round t, when $t-1$ sets have already been picked, and let X_t be the elements already covered by this time. (Hence $X_1 = \emptyset$.) If not all elements have been covered (i.e., if $X_t \neq U$), then use the α-approximate min-ratio oracle with costs $c_t(\cdot)$ and the set X_t to obtain the set $S \in \mathcal{F}$ that approximately minimizes:

$$\frac{\sum_{e \in S \cap R} c_t(e)}{|S \backslash X_t|}.$$

For the analysis, define the potential at the beginning of round t to be

$$\Phi(t) := \sum_{e \in R} (1+\varepsilon)^{A_t(e)/u}.$$

The potential at the beginning of round 1 is $\Phi(1) = |R|$. Observe that the potential is the sum of (some multiple of) the element costs. Our argument will show that because we pick sets with good ratio, the potential does not increase "too fast" and hence is only polynomially large at the end of the algorithm. This will bound each $A_t(e)$ by $O(u \frac{\log N}{\varepsilon})$. Let us now flesh out the details.

LEMMA 4.2. *If we pick set S in round t, then $\Phi(t+1) - \Phi(t) = c_t(S)$.*

PROOF. By the definition of the potential,

$$\begin{aligned}\Phi(t+1) - \Phi(t) &= \sum_{e \in R} (1+\varepsilon)^{A_{t+1}(e)/u} - \sum_{e \in R} (1+\varepsilon)^{A_t(e)/u} \\ &= \sum_{e \in S \cap R} (1+\varepsilon)^{(A_t(e)+1)/u} - (1+\varepsilon)^{A_t(e)/u} = \sum_{e \in S \cap R} (1+\varepsilon)^{A_t(e)/u} \big((1+\varepsilon)^{1/u} - 1\big),\end{aligned}$$

which is $c_t(S)$ by the definition of the round-t cost. □

LEMMA 4.3. *At any round t, there exists a set S with cost-to-coverage ratio $\frac{c_t(S)}{|S \backslash X_t|} \leq \frac{\Phi(t)}{|U \backslash X_t|} \cdot 2\varepsilon$.*

PROOF. Consider an optimal LSC solution $\mathcal{C}^*$. The round-t cost of all the sets in the set cover $\mathcal{C}^*$ is

$$\sum_{S \in \mathcal{C}^*} c_t(S) = \sum_{S \in \mathcal{C}^*} \sum_{e \in S \cap R} (1+\varepsilon)^{A_t(e)/u} \cdot \big((1+\varepsilon)^{1/u} - 1\big) \tag{1}$$

$$\begin{aligned} &= \sum_{e \in R} \left[(1+\varepsilon)^{A_t(e)/u} \cdot \big((1+\varepsilon)^{1/u} - 1\big) \cdot \sum_{S \in \mathcal{C}^* : e \in S} 1 \right] \\ &\leq \sum_{e \in R} (1+\varepsilon)^{A_t(e)/u} \cdot 2\varepsilon/u \cdot u \quad \leq \quad \Phi(t) \cdot 2\varepsilon. \end{aligned} \tag{2}$$

The equality in Equation (1) uses the definition of $c_t(S)$. Inequality (2) used $u \geq 1$ and $\varepsilon \leq 1/4$ and the inequalities $(1+x) \leq e^x$ for all $x \in \mathbb{R}$, and $e^x \leq 1 + 2x$ for $x \in [0,1]$ to get $(1+\varepsilon)^{1/u} \leq e^{\varepsilon/u} \leq 1 + 2\varepsilon/u$.

Since $|U \backslash X_t|$ universe elements are not yet covered, and we could have chosen all the sets in $\mathcal{C}^*$ to cover these remaining elements at cost $\sum_{S \in \mathcal{C}^*} c_t(S)$, there exists some set whose cost-to-coverage-ratio is at most $\Phi(t) \cdot 2\varepsilon / (|U \backslash X_t|)$. □

Let us partition the rounds into phases: Phase i begins in round t if the number of uncovered elements is at most $N/2^i$ for the first time at the beginning of round t. Hence phase 0 begins with round 1 (when the number of uncovered elements is $N/2^0 = N$) and ends at the point we have covered half the elements, and so on. Note that some phases contain no rounds at all.

LEMMA 4.4. *If rounds a and b are the first and last rounds of some phase i, then* $\Phi(b+1) \leq 2 \cdot \Phi(a)$.

PROOF. Consider the beginning of some round $t \in \{a, a+1, \ldots, b\}$ in phase i. By Lemma 4.3, there exists a set with cost-to-coverage-ratio at most $\Phi(t) \cdot 2\varepsilon/(|U \backslash X_t|)$. Moreover, the α-approximate min-ratio oracle finds a set whose cost-to-coverage ratio at most α times as much; that is, at most $2\alpha\varepsilon \frac{\Phi(t)}{|U\backslash X_t|} \leq \frac{1}{4}\frac{\Phi(t)}{|U\backslash X_t|}$, using the definition of $\varepsilon = 1/(8\alpha)$.

Since the potential $\Phi(t)$ is non-decreasing, and $|U \backslash X_t| \geq N/2^{i+1}$ in this phase, this last expression is at most $\frac{1}{4}\frac{\Phi(b+1)}{N/2^{i+1}}$. Moreover, we cover at most $N/2^i$ elements in this phase, so the total cost incurred is at most $\frac{1}{2}\Phi(b+1)$. By Lemma 4.2, the total cost incurred in the phase equals the change in potential, so $\Phi(b+1) - \Phi(a) \leq (1/2) \cdot \Phi(b+1)$. This proves the lemma. □

PROOF (OF THEOREM 4.1). We claim that the potential at the end of the algorithm is at most N^2. Indeed, using Lemma 4.4, the potential at most doubles in each phase, whereas the number of uncovered elements at least halves. Hence, at the end of phase $\log_2 N$, there are strictly less than $N/2^{\log_2 N} = 1$ elements (i.e., no uncovered element) remaining; the potential is at most $2^{\log_2 N} \cdot \Phi(1) = N^2$.

Suppose the last round in which we pick a set is $f - 1$. Then for the final potential to be at most N^2, it must be the case that for each $e \in U$, $(1+\varepsilon)^{A_f(e)/u} \leq N^2$. This means that $A_f(e) \leq u \log_{1+\varepsilon}(N^2) = 2u \frac{\log N}{\log(1+\varepsilon)}$. Since $(1+\varepsilon) \geq e^{\varepsilon/2}$ for $\varepsilon \in [0, 1]$, we get $A_f(e) \leq u \cdot \frac{4}{\varepsilon} \log N$. Substituting $\varepsilon = \frac{1}{8\alpha}$ completes the proof. □

For the problem of approximating the smallest ℓ_∞-norm HL, the previous theorem and Corollary 3.6 imply that an $O(\log n)$-approximation can be computed in $O(n^5)$ time.

5. SIMULTANEOUS ℓ_1- AND ℓ_∞-NORM APPROXIMATION

We can extend the results of Section 4 to find a set cover that simultaneously has small maximum load and small average load. In this case, suppose we are given non-negative values T and u such that there exists a set cover $\mathcal{C}^*$ with $\sum_{e \in R} A_{\mathcal{C}^*}(e) \leq T$ and $A_{\mathcal{C}^*}(e) \leq u$ for all relevant elements $e \in R$. We give an algorithm that finds a set-cover $\mathcal{C}$ such that $\|A_{\mathcal{C}}\|_1 \leq T \cdot O(\alpha \log N)$ and $\|A_{\mathcal{C}}\|_\infty \leq u \cdot O(\alpha \log N)$.

For the algorithm, we use definitions of $c_t(e)$, $c_t(S)$, $\Phi(t)$, and so on, from Section 4, but redefine ε to be $\frac{1}{24\alpha}$. We define the d-cost as $d_e = 1$ for $e \in R$; so $d(S) := |S \cap R|$ for any set S. Note that the c-cost corresponds to minimizing the ℓ_∞-norm (see Section 4) and the d-cost corresponds to minimizing the ℓ_∞-norm.

The algorithm changes as follows: In round t, we now use the α-approximate min-ratio oracle to pick a set S minimizing the "combined" ratio:

$$\frac{c_t(S) + \varepsilon(\Phi(t)/T) \cdot d(S)}{|S \backslash X_t|}. \tag{3}$$

This corresponds to the min-ratio oracle where the cost of each element $e \in R$ is $c_t(e) + \varepsilon\frac{\Phi(t)}{T}$. The scaling factor $\varepsilon\Phi(t)/T$ in the d-costs is used to normalize the ℓ_∞ and ℓ_1 objectives: In particular, we aim to find a solution with total c-cost at most $\varepsilon\Phi(t)$ *and* total d-cost at most T.

For the analysis, consider the beginning of round t, and call a set S *t-light* if $\frac{d(S)}{|S\backslash X_t|} \leq \frac{2T}{|U\backslash X_t|}$ and *t-heavy* otherwise. Let $\mathcal{C}_h^*$ denote the t-heavy sets in $\mathcal{C}^*$ and $\mathcal{C}_l^* := \mathcal{C}^* \backslash \mathcal{C}_h^*$ the t-light sets.

LEMMA 5.1. *The number of elements from $U\backslash X_t$ covered by t-heavy sets in $\mathcal{C}^*$ is at most $\frac{1}{2}|U\backslash X_t|$.*

PROOF. The fraction of the remaining elements that any t-heavy set S covers is $\frac{|S\backslash X_t|}{|U\backslash X_t|} \leq \frac{d(S)}{2T}$. Hence, the total fraction of remaining elements that t-heavy sets in $\mathcal{C}^*$ cover is $\sum_{S\in\mathcal{C}_h^*} \frac{|S\backslash X_t|}{|U\backslash X_t|} \leq \sum_{S\in\mathcal{C}_h^*} \frac{d(S)}{2T} \leq 1/2$. The last inequality is because $\sum_{S\in\mathcal{C}_h^*} d(S) \leq \sum_{S\in\mathcal{C}^*} |S\cap R| \leq T$. □

We can now modify Lemma 4.3 to say the following:

LEMMA 5.2. *At any round t, there exists a t-light set S with cost-to-coverage ratio $\frac{c_t(S)}{|S\backslash X_t|} \leq \frac{\Phi(t)}{|U\backslash X_t|} \cdot 4\varepsilon$.*

PROOF. Let $z := |U\backslash X_t|$ denote the number of universe elements not yet covered. Choosing all t-light sets in $\mathcal{C}^*$ at cost $\sum_{S\in\mathcal{C}_l^*} c_t(S)$, we would cover at least $z/2$ elements (by Lemma 5.1), and hence there exists some set whose cost-to-coverage-ratio is at most

$$\frac{2}{z} \cdot \sum_{S\in\mathcal{C}_l^*} c_t(S) \leq \frac{2}{z} \cdot \sum_{S\in\mathcal{C}^*} c_t(S) \leq \frac{2}{z} \cdot \Phi(t) \cdot 2\varepsilon$$

using the calculations as in Lemma 4.3. □

By Lemma 5.2, we infer that there exists a set S whose combined cost-to-coverage is

$$\frac{c_t(S) + \varepsilon(\Phi(t)/T) \cdot d(S)}{|S\backslash X_t|} \leq \frac{\Phi(t) \cdot 4\varepsilon + \varepsilon(\Phi(t)/T) \cdot 2T}{|U\backslash X_t|} = \frac{\Phi(t) \cdot 6\varepsilon}{|U\backslash X_t|}.$$

Our α-approximate density oracle finds a set S_t with combined cost-to-coverage at most $\alpha \cdot 6\varepsilon \cdot \Phi(t)/|U\backslash X_t| = \frac{1}{4}\Phi(t)/|U\backslash X_t|$. Since the combined cost is a sum of non-negative quantities, we get

$$\frac{c_t(S_t)}{|S_t\backslash X_t|} \leq \frac{1}{4}\frac{\Phi(t)}{|U\backslash X_t|} \qquad \text{and} \qquad \frac{d(S_t)}{|S_t\backslash X_t|} \leq \frac{1}{4\varepsilon}\frac{T}{|U\backslash X_t|}. \tag{4}$$

The bound on $c_t(S_t)/|S_t\backslash X_t|$ can be used in the same fashion as in Section 4 to show that the potential function at most doubles during a phase, and there are at most $\log_2 N$ phases, so the maximum load is at most $O(\alpha \log N) \cdot u$. Moreover, the d-cost incurred in any phase i is at most $(N/2^i) \cdot \frac{1}{4\varepsilon} \cdot \frac{T}{N/2^{i+1}} = \frac{1}{2\varepsilon} \cdot T$. Summing over all $\log_2 N$ phases, and using $\varepsilon = 1/(24\alpha)$, we get the total d-cost is $\sum_{S\in\mathcal{C}} |S\cap R| = \|A_{\mathcal{C}}\|_1 \leq (12\alpha \log_2 N) \cdot T$.

Application to Shortest-Path Labelings. Given an LSC instance arising from the HL problem, and values of u and T, we can use the min-ratio oracles from Section 3 to implement the above algorithm in polynomial time. If the set cover $\mathcal{C}$ we find does not satisfy $\|A_{\mathcal{C}}\|_1 \leq T \cdot O(\alpha \log N)$ or $\|A_{\mathcal{C}}\|_\infty \leq u \cdot O(\alpha \log N)$, then we can infer that the values u, T were not simultaneously feasible. Finally, since any minimal set cover solution has $u \leq N$ and $T \leq N^2$, we can run the algorithm for all $2\log^2 N$ tuples (u, T) where each of u and T are powers of 2. This allows us to return an approximate Pareto-optimal curve: For any tuple (u, T) that is feasible, there is a solution we find with $\|A_{\mathcal{C}}\|_1 \leq T \cdot O(\alpha \log N)$ and $\|A_{\mathcal{C}}\|_\infty \leq u \cdot O(\alpha \log N)$.

6. MINIMIZING ℓ_p-NORMS

We now turn to approximating ℓ_p-norms of the element loads for $p \in [1, \log N]$. We do not consider $p > \log N$ since the ℓ_p-norm of any N-dimensional vector is within a constant factor of its ℓ_∞-norm, when $p \geq \log N$.

THEOREM 6.1. *There is an $O(\alpha \log N)$-approximation algorithm for LSC under any ℓ_p-norm that makes N calls to an α-approximate min-ratio oracle.*

The goal here is to find a set cover $\mathcal{C}$ that minimizes the ℓ_p-norm $\|A_\mathcal{C}\|_p$ of the loads $A_\mathcal{C}$. The algorithm remains similar to the previous ones, where one set is added in each round. In each round t, we define costs on relevant elements as $c_{p,t}(e) := (A_t(e)+1)^p - A_t(e)^p$ for each $e \in R$; so

$$c_{p,t}(S) \quad := \quad \sum_{e \in S \cap R} ((A_t(e)+1)^p - A_t(e)^p). \tag{5}$$

In round t the algorithm picks a set S that (approximately) minimizes $\frac{c_{p,t}(S)}{|S \setminus X_t|}$. Again, X_t is the set of elements covered prior to round t. $A_t(e)$ is the load of element e at the beginning of round t.

The above cost function captures the incremental cost (in the ℓ_p^p-norm objective) of choosing a particular set S in round t. This is a natural choice since we want to minimize the ℓ_p-norm that is equivalent to minimizing its pth power.

For the analysis, let $\mathcal{C}^*$ denote the optimal solution of the given LSC instance and $\mathcal{C}$ the algorithm's solution. To make the analysis easier, we set $A_1(e) = p$ for all relevant elements $e \in R$. The potential function is the following polynomial:

$$\Phi_p(t) := \sum_{e \in R} A_t(e)^p. \tag{6}$$

It immediately follows that if we pick set S_t in round t, $\Phi_p(t+1) - \Phi_p(t) = c_{p,t}(S_t)$. Note that the initial potential is $\Phi_p(1) = |R| \cdot p^p$.

LEMMA 6.2. *For any t, $\sum_{S \in \mathcal{C}^*} c_{p,t}(S) \leq (\mathrm{e}-1) \cdot p \cdot \Phi_p(t)^{(p-1)/p} \cdot \|A_{\mathcal{C}^*}\|_p$.*

PROOF. By the definition of $c_{p,t}(\cdot)$, we know that

$$\sum_{S \in \mathcal{C}^*} c_{p,t}(S) = \sum_{S \in \mathcal{C}^*} \sum_{e \in S \cap R} ((A_t(e)+1)^p - A_t(e)^p) \leq \sum_{S \in \mathcal{C}^*} \sum_{e \in S \cap R} (\mathrm{e}-1)\, p\, A_t(e)^{p-1}$$

(which follows from Lemma 6.3 below and the observation that$A_t(e) \geq A_1(e) \geq p$)

$$\begin{aligned} &= (\mathrm{e}-1)\, p \sum_{e \in R} \left[A_t(e)^{p-1} \sum_{S \in \mathcal{C}^* : e \in S} 1 \right] = (\mathrm{e}-1)\, p \sum_{e \in R} A_t(e)^{p-1} A_{\mathcal{C}^*}(e) \\ &\leq (\mathrm{e}-1)\, p\, \|A_t\|_p^{p-1} \, \|A_{\mathcal{C}^*}\|_p \qquad \text{(by Holder's inequality)} \\ &\leq (\mathrm{e}-1)\, p \cdot \Phi(t)^{(p-1)/p} \cdot \|A_{\mathcal{C}^*}\|_p \end{aligned}$$

Note that we used the fact that we initialized $A_1(e) \geq p$. □

LEMMA 6.3. *For any real $r \geq 1$ and $x \geq r$, $(x+1)^r - x^r \leq (\mathrm{e}-1) r\, x^{r-1}$.*

PROOF. Observe that $(x+1)^r - x^r = x^r((1+x^{-1})^r - 1)$, which equals

$$x^r \sum_{j=1}^{\infty} \binom{r}{j} x^{-j} \leq x^r \sum_{j=1}^{\infty} \frac{r^j}{j!} x^{-j} = x^r \frac{r}{x} \sum_{j=1}^{\infty} \frac{(r/x)^{j-1}}{j!} \leq r\, x^{r-1} \sum_{j=1}^{\infty} \frac{1}{j!} \leq (\mathrm{e}-1) r\, x^{r-1},$$

where we used the inequality $x \geq r$. □

COROLLARY 6.4. *At any round t, there exists a set S with*

$$\frac{c_{p,t}(S)}{|S \setminus X_t|} \le \frac{(\mathrm{e}-1)\cdot p \cdot \Phi_p(t)^{(p-1)/p}\cdot \|A_{\mathcal{C}^*}\|_p}{|U\setminus X_t|}.$$

PROOF. Since all elements in U are covered by $\mathcal{C}^*$, we have $\sum_{S\in\mathcal{C}^*}|S\setminus X_t| \ge |U\setminus X_t|$. The claim now follows by Lemma 6.2 and an averaging argument. □

LEMMA 6.5. *For $\mathcal{C}$ produced by the algorithm, $\|A_{\mathcal{C}}\|_p \le O(\alpha \log N)\cdot \|A_{\mathcal{C}^*}\|_p$.*

PROOF. We define phase $i \in \{0, 1, \dots, \log_2 N\}$ to consist of rounds t where $|U\setminus X_t| \in (\frac{N}{2^{i+1}}, \frac{N}{2^i}]$. Let $\beta := \alpha\,(\mathrm{e}-1)\,p$. Let t^* denote the last round where $\Phi_p(t^*)^{1/p} \le 4\beta\|A_{\mathcal{C}^*}\|_p$, and let i^* denote the phase containing t^*. Note that the starting potential was $\Phi_p(1) = |R|\cdot p^p \le |R|\,\beta^p \le \beta^p\|A_{\mathcal{C}^*}\|_p^p$; hence $t^* \ge 1$.

Consider any phase $i \ge i^*$, and let I (respectively, F) denote the values of Φ_p at the start (respectively, end) of phase i. By our choice of t^* and hence of i^*, it follows that $F^{1/p} \ge \Phi_p(t^*+1)^{1/p} > 4\beta\|A_{\mathcal{C}^*}\|_p$. Moreover, the cost-to-coverage ratio of every set picked in phase i is at most $\frac{\beta\cdot F^{(p-1)/p}\cdot\|A_{\mathcal{C}^*}\|_p}{N/2^{i+1}}$ (using Corollary 6.4) and at most $N/2^i$ elements are covered in this phase. Consequently, the total cost incurred during phase i is at most $2\beta\cdot F^{(p-1)/p}\cdot\|A_{\mathcal{C}^*}\|_p$; moreover, this total cost equals the increase in potential, $F-I$. We can rewrite the resulting inequality as:

$$I^{\frac{1}{p}} \ge F^{\frac{1}{p}}\left(1-\frac{2\beta\|A_{\mathcal{C}^*}\|_p}{F}\right)^{\frac{1}{p}} \ge F^{\frac{1}{p}}\cdot \mathrm{e}^{\frac{-4\beta\|A_{\mathcal{C}^*}\|_p}{p\cdot F^{\frac{1}{p}}}} \ge F^{\frac{1}{p}}\cdot\left(1-\frac{4\beta\|A_{\mathcal{C}^*}\|_p}{p\cdot F^{\frac{1}{p}}}\right).$$

The second inequality above uses $1-x \ge \mathrm{e}^{-2x}$ for $0 \le x \le 1/2$ setting $x = \frac{2\beta\|A_{\mathcal{C}^*}\|_p}{F^{1/p}}$; the final inequality is by $\mathrm{e}^y \ge 1+y$ for all y. This gives us that $F^{1/p} - I^{1/p} \le \frac{4\beta}{p}\cdot\|A_{\mathcal{C}^*}\|_p$ for any phase $i \ge i^*$. Now summing over all such phases $i \ge i^*$ (there are at most $\log_2 N$ of them), we obtain $\Phi_p(\mathsf{final})^{1/p} - \Phi_p(t^*)^{1/p} \le \frac{4\beta}{p}\cdot\log_2 N\cdot\|A_{\mathcal{C}^*}\|_p$. This uses the fact that round t^* lies in phase i^* and $\Phi_p(\cdot)$ is monotone non-decreasing. Finally, using $\Phi_p(t^*)^{1/p} \le 4\beta\|A_{\mathcal{C}^*}\|_p$ and that $\beta = 2\alpha\,(\mathrm{e}-1)\,p$, we have $\Phi_p(\mathsf{final})^{1/p} \le (\frac{4\beta}{p}\log_2 N + 4\beta)\cdot\|A_{\mathcal{C}^*}\|_p = (4\alpha\,(\mathrm{e}-1)\,(\log_2 N + p))\cdot\|A_{\mathcal{C}^*}\|_p$. Since $p \le \log N$, this completes the proof. □

Note that minimizing ℓ_∞ is within constant factors of minimizing $\ell_{\log N}$, so the result subsumes that of Section 4. Moreover, this algorithm does not require us to enumerate over guesses of the optimum load u. The next subsection provides implementation details of our algorithm when applied to hub labeling under ℓ_p-norms and obtains an $O(n^3 p\log n)$ running time.

6.1. Faster Implementation for Hub Labeling via Eager-Lazy Algorithm

A simple runtime analysis of the ℓ_p-norm approximation algorithm for HL is as follows. Each iteration of the algorithm is dominated by n approximate WMDS computations (the min-ratio oracle from Section 3.3) on graphs with n vertices and $O(n^2)$ edges, for a total of $O(n^3)$ time per iteration. Each iteration covers at least one new pair, so the number of iterations is at most $N = O(n^2)$. (Recall that the universe size of the LSC instance is $N = O(n^2)$, corresponding to all vertex-pairs in the HL instance.) So the total running time of the algorithm is $O(n^5)$.

Delling et al. [2014] developed a faster $O(n^3\log n)$ time "*eager-lazy*" $O(\log n)$-approximation algorithm for HL under the ℓ_1-norm. They also sketched an approximation algorithm for HL under ℓ_p-norms with $O(n^3\log n\min(p,\log n))$ runtime. For completeness, we outline the latter algorithm here.

We first describe a new procedure (called *α-eager evaluation*) for WMDS. This is similar to the 2-approximate WMDS algorithm of Section 3.3. Fix a constant $\alpha > 1$. Consider any WMDS instance: a graph $G = (V, E)$ with vertex-weights $c : V \to \mathcal{R}_+$, and let μ be an upper bound on its WMDS value. The procedure maintains a current graph G'; initially $G' = G$. Let $\delta'(v)$ denote the degree of v in G'. While the weighted density of G' is less than $\mu/(2\alpha)$, the procedure deletes a vertex v that minimizes $\delta'(v)/c(v)$. Let $G'' = (V'', E'')$ be the subgraph (possibly empty) that remains at the end of this procedure. Let $\tilde{G}$ be the graph G with all edges in E'' deleted; note that each edge of $\tilde{G}$ is incident to some vertex of $V \setminus V''$.

LEMMA 6.6. *(1) If G'' is not empty, then its density is at least $\mu/(2\alpha)$ and (2) the WMDS value of $\tilde{G}$ is less than μ/α.*

PROOF. By construction, we stop when the density of the current graph G' is at least $\mu/(2\alpha)$, which implies condition (1) for the final graph G''.

For any vertex $v \in V \setminus V''$ (which is deleted in the above procedure), let m_v be the number of edges adjacent to v when it was deleted from G'. We show that $m_v/c(v) < \mu/\alpha$. Recall that the procedure chooses v to minimize $\delta'(v)/c(v)$ over all vertices in the current graph G'. Therefore $m_v/c(v) \geq \mu/\alpha$ would imply the following bound on the density of G':

$$\frac{|E(G')|}{c(V')} \geq \frac{\sum_{v \in V'} \delta'(v)}{2c(V')} \geq \frac{\sum_{v \in V'} \mu \cdot c(v)/\alpha}{2c(V')} \geq \frac{\mu}{2\alpha}.$$

But then the algorithm would terminate before deleting v.

Let $G^* = (V^*, E^*)$ be a WMDS of $\tilde{G}$. Since all edges of $\tilde{G}$ are incident to some vertex in $V \setminus V''$, we have $|E^*| \leq \sum_{V^* \setminus V''} m_v$. So

$$|E^*| < \sum_{V^*} \mu \cdot c(v)/\alpha = \mu \cdot c(V^*)/\alpha.$$

Thus the density of G^* is less then μ/α, proving condition (2). □

Note that G'' may be empty, in which case (μ/α) is an improved upper bound on the WMDS value of $G = \tilde{G}$.

As for the algorithm of Section 3.3, the time complexity of this α-eager procedure for WMDS is $O(m + n \log n) = O(n^2)$.

Next we discuss a more efficient implementation of the ℓ_p-norm HL algorithm, using *lazy evaluation* [Cohen et al. 2003; Schenkel et al. 2004] and the above α-eager procedure for WMDS. The ℓ_p-norm HL algorithm is iterative, where in each iteration it invokes the min-ratio oracle. Recall that the min-ratio oracle (Section 3.2) involves WMDS instances on the center graphs $\{G_v\}_{v \in V}$. The vertex-weights to these WMDS instances are as defined in the ℓ_p-norm LSC algorithm. Note that the WMDS values of the center graphs are monotonically non-increasing over the iterations: This is because edges can only be deleted from these graphs, and vertex-weights can only increase. Therefore, an upper bound on WMDS of a center graph computed during any iteration of the algorithm remains valid in all subsequent iterations.

For a fixed constant $\alpha > 1$, the α-eager lazy algorithm for ℓ_p-norm HL works as follows. It maintains a partial labeling (initially empty) and the set P of uncovered pairs (initially P contains all pairs of vertices). During initialization, the algorithm also computes upper bounds μ_v on the WMDS values of each center graph G_v using the 2-approximation algorithm (Section 3.3) and puts these values into a max-heap Q. At each iteration, the algorithm extracts the maximum μ_v from the heap and builds G_v (using the uncovered pairs P as described in Section 3.2). Then the algorithm applies the α-eager evaluation for WMDS on G_v using the upper bound μ_v. If the evaluation finds a non-empty subgraph G'', then the algorithm adds v to the labels of the vertices of G'' and updates P. Then the algorithm sets $\mu_v = \mu_v/\alpha$ and updates this in the priority queue. Using data structures described in Delling et al. [2014], the time taken in each iteration is $O(n^2)$. The algorithm terminates when P becomes empty.

To bound the number of iterations of the algorithm, we note that if v is selected at an iteration, then μ_v decreases by a factor of at least α. The vertex-weights are always between 1 and $O(n^p)$, so the density of any non-empty graph is between $\Omega(n^{-p})$ and $O(n)$. It follows that each center graph G_v is selected only in $O(p \log n)$ iterations. Hence the total number of iterations is $O(np \log n)$.

The above analysis gives an $O(n^3 p \log n)$ bound on the running time of the algorithm. The $O(n^3 \log n \min(p, \log n))$ time bound follows since we may assume $p = O(\log n)$ without loss of generality.

7. MULTIPLE NORMS SIMULTANEOUSLY

The approach of the previous section naturally extends to give solutions that are good with respect to multiple ℓ_p-norms; we now state the changes required in the algorithm and analysis for two norms. Specifically, given $p, q \in [1, \log N]$ and values P, Q, we want to find (if possible) a set-cover $\mathcal{C}$ with $\|A_{\mathcal{C}}\|_p \le O(\alpha \log N) \cdot P$ and $\|A_{\mathcal{C}}\|_q \le O(\alpha \log N) \cdot Q$.

We describe the algorithm assuming that there is some set-cover $\mathcal{C}^*$ such that $\|A_{\mathcal{C}^*}\|_p \le P$ and $\|A_{\mathcal{C}^*}\|_q \le Q$. In this case, our algorithm will find a solution $\mathcal{C}$ with guarantees as above. (If the algorithm does not find such a solution, then we obtain a certificate that no set-cover $\mathcal{C}'$ has $\|A_{\mathcal{C}'}\|_p \le P$ and $\|A_{\mathcal{C}'}\|_q \le Q$.)

The algorithm is again iterative and chooses one set in each iteration. The round-t costs $c_{p,t}$ and $c_{q,t}$, and the potentials $\Phi_p(t)$ and $\Phi_q(t)$ are defined as in Equations (5) and (6). We initialize the element loads $A_1(e)$ to $\max\{p, q\} \le \log N$. In each round t the algorithm picks the set S minimizing the "combined" ratio:

$$\frac{1}{|S \backslash X_t|} \cdot \left(\frac{c_{p,t}(S)}{p \cdot \Phi_p(t)^{(p-1)/p} \cdot P} + \frac{c_{q,t}(S)}{q \cdot \Phi_q(t)^{(q-1)/q} \cdot Q} \right). \tag{7}$$

This cost function normalizes the ℓ_p- and ℓ_q-norm objectives and combines them into a single objective that is convenient to work with.

LEMMA 7.1. *At any round t, there exists a set S with ratio (7) at most* $2(\mathrm{e}-1)/|U \backslash X_t|$.

PROOF. Using Lemma 6.2, we obtain $\sum_{S \in \mathcal{C}_r^*} c_{r,t}(S) \le (\mathrm{e}-1) \cdot r \cdot \Phi_r(t)^{(r-1)/r} \cdot \|A_{\mathcal{C}^*}\|_r$ for $r = p, q$. Since $\|A_{\mathcal{C}^*}\|_p \le P$ and $\|A_{\mathcal{C}^*}\|_q \le Q$, we have

$$\sum_{S \in \mathcal{C}_r^*} \left(\frac{c_{p,t}(S)}{p \cdot \Phi_p(t)^{(p-1)/p} \cdot P} + \frac{c_{q,t}(S)}{q \cdot \Phi_q(t)^{(q-1)/q} \cdot Q} \right) \le 2(\mathrm{e}-1).$$

Moreover, as all elements in U are covered by $\mathcal{C}^*$, we have $\sum_{S \in \mathcal{C}^*} |S \backslash X_t| \ge |U \backslash X_t|$. The lemma now follows by an averaging argument. □

Based on this lemma and the α-approximation min-ratio oracle, the set S_t picked by the algorithm has ratio (7) at most $\alpha \cdot \frac{2(\mathrm{e}-1)}{|U \backslash X_t|}$. Finally, the analysis from Lemma 6.5 carries over virtually unchanged for $r = p, q$, the only difference being the definition $\beta := 2(\mathrm{e}-1)\alpha r$.

The above algorithm extends to finding LSC that is within an $O(\alpha k \log N)$ factor of k different targets with respect to k different ℓ_p-norms $p_1, p_2, \ldots, p_k$.

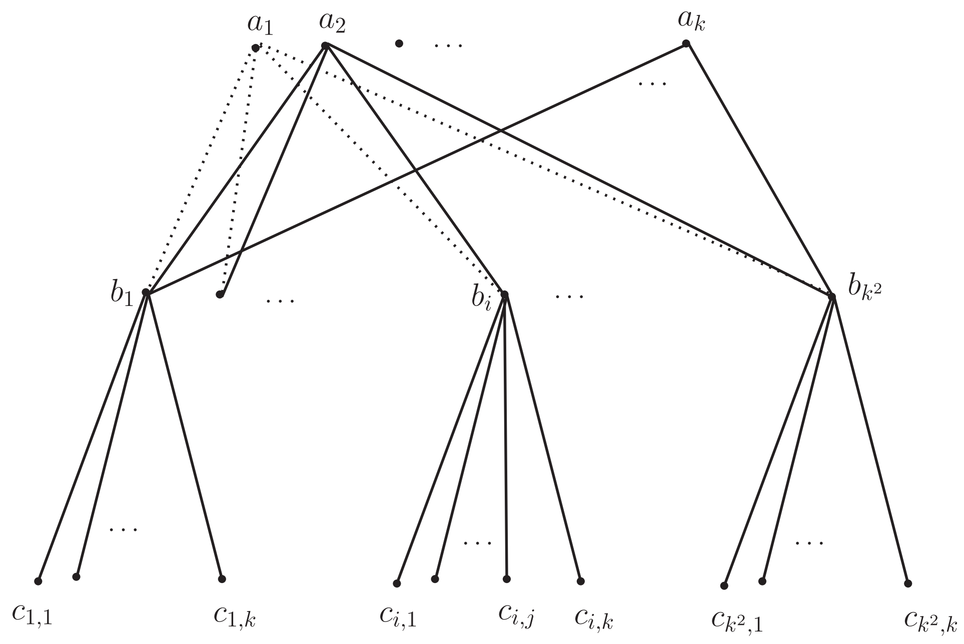

Fig. 1. The graph G in the example. The dotted edges (those adjacent to a_1 have length $1-\varepsilon$, all other edges have length 1.

THEOREM 7.2. *There is an algorithm that for any instance of LSC with k ℓ_p-norms $p_1, p_2, \ldots, p_k \in [1, \log N]$ and values $T_1, T_2, \ldots, T_k$, outputs one of the following:*

—*a set-cover $\mathcal{C}$ with $\|A_{\mathcal{C}}\|_{p_j} \leq O(\alpha k \log N) \cdot T_j$ for all $j \in \{1, 2, \ldots, k\}$, or*
—*a certificate that no set-cover $\mathcal{C}'$ has $\|A_{\mathcal{C}'}\|_{p_j} \leq T_j$ for all $j \in \{1, 2, \ldots, k\}$.*

This algorithm makes at most N calls to an α-approximate min-ratio oracle.

7.1. Non-Existence of Simultaneous Optimality

The results of the previous section imply that if there existed a labeling with total label size T and maximum label size M, then the algorithm of Section 7 would find a labeling with total label size $O(T \log N)$ and maximum label size $O(M \log N)$. A natural question is whether we can hope for a stronger guarantee: to find a labeling whose total label size is within a logarithmic factor of the smallest possible total label size T^*, and whose maximum label size is within a logarithmic factor of the smallest possible maximum label size M^*.

In this section, we show this is not possible. Specifically, we show an example such that that for any labeling with total size T and maximum size M, either T is polynomially bigger than T^* or M is polynomially bigger than M^*.

For an integer parameter k, the undirected graph G has three sets of vertices, $A = \{a_1, a_2, \ldots, a_k\}$, $B = \{b_1, b_2, \ldots b_{k^2}\}$, and $C = \{c_{ij} \mid i \in [k^2], j \in [k]\}$, of size k, k^2, and k^3, respectively. Below, for any integer t, we denote $[t] := \{1, 2, \ldots, t\}$. Every vertex in A is connected to all vertices in B. Vertices in C are partitioned into k^2 groups of size k each: For each $i \in [k^2]$, group $C_i = \{c_{ij} \mid j \in [k]\}$ corresponds to the vertex $b_i \in B$. Each vertex in $b_i \in B$ is connected to all k vertices in its group C_i. There are no other edges in the graph. All edges have length 1, except for the edges from a_1 to B, which are of length $1 - \varepsilon$ (for some $0 < \varepsilon < 1$). The total number of vertices of the graph is $n = \Theta(k^3)$. See also Figure 1.

Observe that the shortest paths in graph G are as follows:

—For any pair $a, a' \in A$ of vertices, the path a, b, a' is a shortest a-a' path for every $b \in B$. For any vertex $a \in A$ and $b \in B$, the edge (a, b) is the unique shortest path. For any $a \in A$ and $c \in C_i$, the path a, b_i, c is the unique shortest path.
—For vertices $b, b' \in B$, the path b, a_1, b' is the unique shortest path. For any $b_i \in B$ and $c \in C$, the unique shortest path between vertex b_i and c is (i) the edge (b_i, c) if $c \in C_i$, or (ii) the path $b_i, a_1, b_{i'}, c$ if $c \in C_{i'}$ (for $i' \neq i$).
—For vertices c, c' in the same group C_i, the path c, b_i, c' is the unique shortest path. For $c \in C_i$ and $c' \in C_{i'}$ (for $i' \neq i$), the path $c, b_i, a_1, b_{i'}, c'$ is the unique shortest path.

Bounding Total Label Size. Graph G has a labeling L_t of total size $O(k^3)$ as follows:

$$L_t(v) = \begin{cases} \{a\} \cup B & \text{if } v = a \in A \\ \{b, a_1\} & \text{if } v = b \in B \\ \{c, b_i, a_1\} & \text{if } v = c \in C_i,\ i \in [k^2]. \end{cases}$$

It can be checked directly that this is a valid labeling, that is, for each pair u, v of vertices some vertex on a shortest $u - v$ path lies in $L_t(u) \cap L_t(v)$. The total label size is $k \cdot (k^2 + 1) + 2 \cdot k^2 + 3 \cdot k^3 = O(k^3)$; so $T^* = O(k^3)$.

Bounding Maximum Label Size. On the other hand, there is a different labeling L_m for G with maximum label size $O(k)$,

$$L_m(v) = \begin{cases} A & \text{if } v = a \in A \\ \{b\} \cup A & \text{if } v = b \in B \\ \{c, b_i\} \cup A & \text{if } v = c \in C_i,\ i \in [k^2]. \end{cases}$$

Again, it can be checked directly that this is a valid labeling. So $M^* = O(k)$.

Now consider an arbitrary labeling L with total size T and the maximum size M. Fix any vertex $a \in A$ and consider the shortest paths from a to all vertices in C. For any $v \in L(a)\backslash\{a\}$, the number of vertices $c \in C$ for which an $a - c$ shortest path contains v is at most k. So the labels of at least $k^3 - k|L(a)| \geq k^3 - kM$ vertices in C must contain a. Since this holds for every $a \in A$, we have $T \geq k(k^3 - kM)$, that is, $T + k^2 M \geq k^4$. Hence, if $T = o(k^4)$, then $M = \Omega(k^2)$, and if $M = o(k^2)$, then $T = \Omega(k^4)$. Therefore either T/T^* or M/M^* is $\Omega(n^{1/3})$.

ACKNOWLEDGMENTS

We thank Amos Fiat and Haim Kaplan for interesting discussions. A. Gupta thanks the (erstwhile) Microsoft Research Silicon Valley laboratory, as well as IBM T.J. Watson Research Center and the IEOR Department at Columbia University for their generous hospitality.

REFERENCES

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato Fonseca F. Werneck. 2011. A hub-based labeling algorithm for shortest paths in road networks. In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*. 230–241.

Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato Fonseca F. Werneck. 2012. Hierarchical hub labelings for shortest paths. In *Proceedings of the 20th Annual European Symposium on Algorithms (ESA'12)*. 24–35.

James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. 1997. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM* 44, 3 (1997), 486–504.

Baruch Awerbuch, Yossi Azar, Edward F. Grove, Ming-Yang Kao, P. Krishnan, and Jeffrey Scott Vitter. 1995. Load balancing in the L_p norm. In *36th Annual Symposium on Foundations of Computer Science*. 383–391.

Yossi Azar and Amir Epstein. 2005. Convex programming for scheduling unrelated parallel machines. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*. 331–337.

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

Vašek Chvátal. 1979. A greedy heuristic for the set-covering problem. *Math. Operat. Res.* 4 (1979), 233–235.

Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. 2003. Reachability and distance queries via 2-hop labels. *SIAM J. Comput.* 32, 5 (2003), 1338–1355.

Daniel Delling, Andrew V. Goldberg, Ruslan Savchenko, and Renato F. Werneck. 2014. Hub labels: Theory and practice. In *Proceedings of the 13th International Symposium on Experimental Algorithms (SEA'14)*. 259–270.

Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. 2009. Engineering route planning algorithms. In *Algorithmics of Large and Complex Networks*, Dorothea Wagner, Jürgen Lerner, and Katharina Zweig (Eds.). 117–139.

Uriel Feige. 1998. A threshold of ln n for approximating set cover. *J. ACM* 45, 4 (1998), 634–652.

Michael L. Fredman and Robert Endre Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* 34, 3 (1987), 596–615.

Giorgio Gallo, Michael D. Grigoriadis, and Robert Endre Tarjan. 1989. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* 18, 1 (1989), 30–55.

Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. 2004. Distance labeling in graphs. *J. Algor.* 53, 1 (2004), 85–112.

Andrew V. Goldberg. 2008. A practical shortest path algorithm with linear expected time. *SIAM J. Comput.* 37, 5 (2008), 1637–1655.

Guy Kortsarz and David Peleg. 1994. Generating sparse 2-spanners. *J. Algor.* 17, 2 (1994), 222–236.

V. S. Anil Kumar, Madhav V. Marathe, Srinivasan Parthasarathy, and Aravind Srinivasan. 2009. A unified approach to scheduling on unrelated parallel machines. *J. ACM* 56, 5 (2009).

Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. 2015. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015)*. 1049–1065.

Ralf Schenkel, Anja Theobald, and Gerhard Weikum. 2004. HOPI: An efficient connection index for complex XML document collections. In *Advances in Database Technology—Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004)*. 237–255.

Mikkel Thorup. 1999. Undirected single-source shortest paths with positive integer weights in linear time. *J. ACM* 46, 3 (1999), 362–394.

Received August 2015; revised September 2016; accepted September 2016