AN IMPROVED APPROXIMATION ALGORITHM FOR K-MEDIAN

NEAL E. YOUNG*

Abstract. We give a polynomial-time approximation algorithm for the (not necessarily metric) k-Median problem. The algorithm is an *α-size-approximation* algorithm for $\alpha < 1 + 2\ln(n/k)$. That is, it guarantees a solution having size at most $\alpha \times k$, and cost at most the cost of any size-k solution. This is the first polynomial-time approximation algorithm to match the well-known bounds of H_Δ and $1 + \ln(n/\mathsf{opt})$ for unweighted Set Cover (a special case) within a constant factor. It matches these bounds within a factor of 2. The algorithm runs in time $O(k\,m\log(n/k)\log m)$, where n is the number of customers and m is the instance size.

1. Introduction. An instance of the *k-Median* problem is given by an edge-weighted bipartite graph $G = (U, W, E)$, where U is the set of *centers*, W is the set of *customers*, and each center/customer pair $(i, j) \in E \subseteq U \times W$ has an associated *cost* $c_{ij} \geq 0$, which we interpret as the cost of assigning customer i to center j. The goal is to choose a set $C \subseteq U$ of k centers of minimum *cost*, defined to be $c(C) = \sum_{j\in W} \min_{i\in C} c_{ij}$, with the interpretation that each customer is assigned to its closest center in C. Let $n = |W|$ and $m = |E|$.

An *α-size-approximate solution* is a set $C \subseteq W$ of size at most $\alpha\,k$ and cost at most the minimum cost of any size-k solution. An algorithm that guarantees such a solution is an *α-size-approximation algorithm*.

The restriction of k-Median to zero-cost instances is equivalent to the well-studied unweighted Set Cover problem—the Set Cover instance admits a cover of size k if and only if the corresponding k-Median instance has a size-k solution of cost zero. Assuming P≠NP, this implies that no polynomial-time approximation algorithm for k-Median guarantees solutions of size $(1-\epsilon)\ln n$ (for any constant $\epsilon > 0$) with cost approximating the optimum within any finite factor [3]. This motivates the study of polynomial-time bicriteria-approximation algorithms.

The first such algorithm, by Lin and Vitter, produces solutions of size at most $(1 + 1/\epsilon)(1 + \ln n)k$ and cost at most $1 + \epsilon$ times the minimum cost of any size-k solution [5]. Here $\epsilon > 0$ is an input parameter that controls the tradeoff between size and cost. The second algorithm, by Young, improves this tradeoff by producing solutions of size at most $(1 + \ln(n + n/\epsilon))k$ and cost at most $1 + \epsilon$ times the minimum cost of any size-k solution [9]. The third, by Chrobak et al., incurs no tradeoff: it guarantees solutions of size $O(\log n)$ and cost at most the minimum cost of any size-k solution [1]. That is, it is an $O(\log n)$-size-approximation algorithm as defined above.

For the special case of unweighted Set Cover, stronger bounds are known in terms of k and the maximum set size Δ. Johnson [4] and Lovasz [6] show that the greedy algorithm has approximation ratio at most $H_\Delta = \sum_{h=1}^{\Delta} 1/h$, where $\Delta \leq n$ is the maximum set size. (Chvatal [2] extends this to weighted Set Cover.) A folklore result (see Section 3) is that the ratio is at most $1 + \ln(n/\textsc{opt})$, where $\textsc{opt}$ is the optimal cover size. Note that $n/\textsc{opt} \leq \Delta$ and $\gamma + \ln\Delta \leq H_\Delta$ where $\gamma = 0.577..$ is Euler's constant, so this bound can be smaller than H_Δ but never exceeds it by more than $1 - \gamma$. (Slavík shows a bound of $\ln(n/\ln n) + O(1)$, which is asymptotically stronger when $\textsc{opt} = o(\log n)$ [7].)

Our main result (Corollary 4.2) is a polynomial-time α-size-approximation algorithm for k-Medians, where $\alpha \leq 1 + 2\ln(n/k) \leq 2H_\Delta$. (For k-Median $\Delta = \max\left\{|\{j \in$

* University of California, Riverside

minimize $c \cdot y = \sum_{ij} c_{ij}\, y_{ij}$		maximize $-k\,\mu + \sum_j \delta_j$	
$(\forall i \in U,\, j \in W)$	$x_i \geq y_{ij} \geq 0$	$(\forall i \in U,\, j \in W)$	$\pi_{ij} \geq 0$
$(\forall j \in W)$	$\sum_i y_{ij} = 1$	$(\forall i \in U,\, j \in W)$	$\pi_{ij} \geq \delta_j - c_{ij}$
	$\sum_i x_i = k$	$(\forall i \in U)$	$\mu \geq \sum_j \pi_{ij}$

FIG. 1.1. *The standard k-Median linear-program relaxation and its dual.*

$W : (i,j) \in E\}| : i \in U\big\}$ is the maximum number of customers that any center can serve. Note $n/k \leq \Delta$ for any feasible instance.) This matches the Set Cover bounds H_Δ and $1 + \ln(n/\text{OPT})$ within a factor of two. No previous result matched either bound within any constant factor.

The only previous polynomial-time size-approximation algorithm [1] requires solving the standard linear-program (LP) relaxation for k-Median (Figure 1.1). Our algorithm avoids that. It runs in time $O(k\,m \log(n/k) \log m)$.

2. Preliminaries. Let $\mathbb{R}_0 = \{x \in \mathbb{R} : x \geq 0\}$ and $\overline{\mathbb{R}}_0 = \mathbb{R}_0 \cup \{\infty\}$.

Fix a k-Median instance $G = (U, W, E)$. For $(i,j) \in E$, let $c_{ij} \in \mathbb{R}_0$ denote the cost of assigning customer j to center i. To ease notation, take $c_{ij} = \infty$ for $(i,j) \in (U \times W) \setminus E$.

Fix $T = \lceil k \ln(n^2/(2k(2k+1))) \rceil$ and $\alpha_{kn} = T/k + 2$, and let $\lambda^* \in \overline{\mathbb{R}}_0$ denote the optimum cost of the standard LP relaxation (Figure 1.1). The algorithms here find solutions of cost at most λ^* and size at most $T + 2k = \alpha_{kn}\, k$. These are α_{kn}-size-approximate solutions, because any size-k solution has cost at least λ^*.

Assume without loss of generality that $2 \leq k \leq n/3$. (If $k = 1$, the optimal size-1 solution $C = \{\arg\min_{i \in U} \sum_{j \in W} c_{ij}\}$ can be computed in linear time. If $k > n/3$, the size-n solution $C = \{\arg\min_{i \in U} c_{ij} : j \in W\}$ has minimum possible cost and size $n \leq \alpha_{nk}\, k$.)

Assume without loss of generality that every customer $j \in W$ has a center $i \in U$ such that $c_{ij} = 0$. (Otherwise, for each customer j, subtract $\min_{i' \in U} c_{i'j}$ from each c_{ij}. This reduces the cost of each solution by the same non-negative amount, $\sum_{j \in W} \min_{i \in U} c_{ij}$, so any given solution achieves optimal cost for the original instance if and only if it does so for the modified instance.)

LEMMA 2.1. $\alpha_{kn} < 2\ln(n/k) + 2 - 2\ln 2 + 1/(2k) + 1/(4k^2) < 1 + 2\ln(n/k) < 2H_\Delta$

The proof of the lemma is in Appendix A.

The capped cost, defined next, plays a central role in the algorithms. Section 5 gives some intuition for its definition. Recall $k \leq n/3$.

DEFINITION 2.2 (capped cost). *Given $\lambda \in \overline{\mathbb{R}}_0$ and a set $C \subseteq W$ of centers, define the* capped cost *of C to be $\tilde{c}(\lambda, C) = \sum_{j \in W} \tilde{c}_j(\lambda, C)$, where*

$$\tilde{c}_j(\lambda, C) = \min\Big(1/(2k+1), (1 - 2k/n) \min_{i \in C} c_{ij}/\lambda\Big)$$

is the capped cost of customer j *(with respect to C). In this context we interpret $0/0$ and ∞/∞ as zero.*

We'll use the following utility lemma to work with capped costs. It shows that the addition of a single center i' to a given set C of customers can always decrease the capped cost by a certain amount. The lemma implicitly gives a lower bound on λ^*.

$\mathbb{R}_0$ – the non-negative reals	$\overline{\mathbb{R}}_0$ – $\mathbb{R}_0 \cup \{\infty\}$
c – costs for k-Median instance	i, j – center $i \in U$, customer $j \in W$
λ^* – optimum LP cost	n, m – $n = \lvert W\rvert$, $m = \lvert E\rvert$
λ – capped-cost parameter	C – set of centers
$\tilde{c}(\lambda, C)$ – capped cost of C	$\tilde{c}_j(\lambda, C)$ – capped cost of customer j (w.r.t. C)
T – number of centers from first phase: $T = \lceil k \ln(n^2/(2k(2k+1))) \rceil$	
α_{kn} – size-approximation ratio:	$\alpha_{kn} = T/k + 2$

TABLE 2.1
Notation

LEMMA 2.3. *For any $\lambda \in \overline{\mathbb{R}}_0$ and $C \subseteq W$,*

$$\min_{i \in U} \tilde{c}(\lambda, C \cup \{i'\}) \leq (1 - 1/k)\tilde{c}(\lambda, C) + (1/k)(1 - 2k/n)\lambda^*/\lambda.$$

We prove the lemma in the appendix. For intuition consider the case that the LP admits an optimal integer solution, that is, a set $C^* \subseteq W$ of k centers with cost $c(C^*) = \lambda^*$. Adding all k centers from C^* to C would reduce the capped cost to at most $\tilde{c}(\lambda, C^*)$. So, observing that $\tilde{c}(\lambda, C)$ (for fixed λ) is a submodular function, there exists a center $i \in U$ to add that reduces the capped cost by at least $(1/k)(\tilde{c}(\lambda, C) - \tilde{c}(\lambda, C^*))$. With $\tilde{c}(\lambda, C^*) \leq (1 - 2k/n)c(C^*)/\lambda = (1 - 2k/n)\lambda^*/\lambda$ and some algebra, this implies the lemma. The full proof considers the optimal LP solution (x^*, y^*) instead of C^*.

3. Slow algorithm. This section describes a slow algorithm. Section 4 will build on it to prove the main result. The approach is similar in spirit to the folklore result that, for unweighted Set Cover, the greedy algorithm gives $(1 + \ln(n/k))$-approximation, where $k = \text{OPT}$ is the minimum set-cover size. This can be shown as follows: each set chosen by the greedy algorithm covers at least a $1/k$ fraction of the remaining elements, so after any iteration t at most $n(1 - 1/k)^t < ne^{-t/k}$ elements remain uncovered. In particular, after $t = \lceil k \ln(n/k) \rceil$ iterations less than $ne^{-t/k} = k$ elements remain. Each subsequent iteration covers at least 1 element, so there are at most $k - 1$ additional iterations, for a total of at most $(\ln(n/k) + 1)k$.

Similarly, the k-Median algorithms here have two phases. The first phase chooses up to $T = \lceil k \ln(n^2/(2k(2k+1))) \rceil$ centers greedily, minimizing the capped cost with each choice. We show (using Lemma 2.3) that this generates a set C whose capped cost is strictly less than 1. The second phase then "polishes" the partial solution, adding up to $2k$ additional centers, each chosen greedily to reduce the capped cost of a particular customer to zero. This suffices to obtain a full solution (of size at most $T + 2k$) whose true cost is at most the optimum LP cost λ^*.

The goal of the first phase is to compute a set C of at most T centers with capped cost $\tilde{c}(\lambda^*, C)$ strictly less than 1:

THEOREM 3.1 (first phase). *A set $C \subseteq W$ of size at most T with $\tilde{c}(\lambda^*, C) < 1$ can be computed in polynomial time.*

Proof. The algorithm solves the LP to obtain λ^*, then chooses centers greedily, in each step decreasing the capped cost $\tilde{c}(\lambda^*, C)$ as much as possible, just until the capped cost is less than 1:

1. solve the LP to obtain the optimum cost λ^*
2. let $C \leftarrow \emptyset$
3. while $\tilde{c}(\lambda^*, C) \geq 1$:

4. let $i' = \arg\min_{i \in U} \tilde{c}(\lambda^*, C \cup \{i\})$
5. let $C \leftarrow C \cup \{i'\}$
6. return C

Let C_t denote the set C at the end of each iteration t. Let $C_0 = \emptyset$. Let $p = 1 - 1/k$. From Lemma 2.3 (with $\lambda = \lambda^*$), and using $\tilde{c}(\lambda, \emptyset) = n/(2k+1)$, it follows inductively that the algorithm maintains the invariant

$$\tilde{c}(\lambda^*, C_t) \le p^t \, n/(2k+1) + (1 - p^t)(1 - 2k/n). \tag{3.1}$$

Thus, the loop cannot iterate more than $T = \lceil k \ln(n^2/(2k(2k+1))) \rceil$ times, because if it reaches iteration T, then, after that iteration, using $p^T < \exp(-T/k) \le 2k(k+1)/n^2$, the invariant implies

$$\tilde{c}(\lambda^*, C_T) < \frac{2k(2k+1)}{n^2} \times \frac{n}{2k+1} + 1 - \frac{2k}{n} = 1. \qquad \square$$

The first phase (by Theorem 3.1) computes a set $C \subseteq W$ of size at most T such that the capped cost $\tilde{c}(\lambda^*, C)$ is strictly less than 1. Given this C, the second phase computes the desired α_{nk}-size-approximate solution:

THEOREM 3.2 (second phase). *Given any set $C \subseteq U$ of at most T centers, and $\lambda \in \bar{\mathbb{R}}_0$ such that $\tilde{c}(\lambda, C) < 1$, a set $C' \subseteq U$ of size at most $T + 2k$ and cost at most λ can be computed in time $O(m + n \log n)$.*

COROLLARY 3.3. *K-Median admits a polynomial-time α_{nk}-size-approximation algorithm.* ∎

Before we prove the theorem, note for intuition that C can have at most $2k$ customers with $\tilde{c}_j(\lambda, C) = 1/(2k+1)$, simply because each unassigned customer contributes $1/(2k+1)$ to $\tilde{c}(\lambda, C)$, which is less than 1. By adding one center i with $c_{ij} = 0$ for each unassigned customer (thereby adding at most $2k$ centers total) we could assure that all customers are assigned. Similarly, by adding such centers for the $2k$ customers with maximum capped cost, we could reduce the capped cost by a factor of $1 - 2k/n$, to less than $1 - 2k/n$, which (by inspection of the capped cost) is just enough to ensure that the true cost is at most λ. Naively, accomplishing both of above goals would take $4k$ additional centers. The theorem shows that to accomplish both goals it suffices to add just $2k$ centers.

Proof of Theorem 3.2. Given C, compute C' as follows:

1. while $c(C) > \lambda$:
2. let $j = \arg\max_{j \in W} \tilde{c}_j(\lambda, C)$
3. let $C \leftarrow C \cup \{i'\}$ where $i' = \arg\min_{i \in U} c_{ij}$ — *note: now* $c_{i'j} = \tilde{c}_j(\lambda, C) = 0$.
4. return C

This can be done in $O(m + n \log n)$ time by presorting the set W of customers by decreasing $\tilde{c}(\lambda, C)$. To finish, we show that the loop iterates at most $2k$ times. Assume it iterates at least $2k$ times (otherwise we are done).

OBSERVATION 3.1. *For any vector $b \in \mathbb{R}_0^n$ such that $\sum_j b_j < 1$, let b' be b with its $2k$ largest values replaced by zero. Then*

$$\max_{j \in W} b'_j < 1/(2k+1), \quad \textit{and} \quad \sum_{j \in W} b'_j < 1 - 2k/n.$$

greedy$'(c)$ — *input: cost vector* c

1. let $C = \emptyset$ and $\lambda_0 = 0$ and $t \leftarrow 0$
2. while $\tilde{c}(\lambda_t, C) \geq 1$:
3. let $t \leftarrow t + 1$
4. let $\tau \leftarrow (1 - 1/k)\tilde{c}(\lambda_{t-1}, C) + (1/k)(1 - 2k/n)$
5. let $\lambda_t = \min\left\{\lambda \geq \lambda_{t-1} : \min_{i \in U} \tilde{c}(\lambda, C \cup \{i\}) \leq \tau\right\}$
6. choose i' such that $\tilde{c}(\lambda_t, C \cup \{i'\}) \leq \tau$
7. let $C \leftarrow C \cup \{i'\}$
8. return (λ_t, C)

FIG. 4.1. *Faster replacement for first phase.*

(Indeed, the maximum value in b' is the minimum of the $2k+1$ largest values in b, which is at most the average of those values, which is at most $\sum_{b=1}^{n} b/(2k+1) < 1/(2k+1)$. So $\max_j b'_j$ is bounded as claimed. Zeroing $2k$ random values in b would decrease its sum by a factor of $1 - 2k/n$ in expectation. Zeroing the $2k$ largest values decreases its sum by at least that. So $\sum_j b'_j$ is bounded as claimed.)

OBSERVATION 3.2. *Let C_{2k} be C after $2k$ iterations of the loop. Then*

$$\max_{j \in W} \tilde{c}_j(\lambda, C_{2k}) < 1/(2k+1), \text{ and } \tilde{c}(\lambda, C_{2k}) < 1 - 2k/n.$$

(To see this, let C_0 refer to C as given, before the loop executes, and define b in $\overline{\mathbb{R}}_0^n$ by $b_j = \tilde{c}_j(\lambda, C_0)$. Let b' be obtained from b by replacing its $2k$ largest values by zero. By the definition of C_{2k}, we have $\tilde{c}_j(\lambda, C_{2k}) \leq b'_j$. The observation follows follows from Observation 3.1, along with $\sum_j b_j = \tilde{c}(\lambda, C_0) < 1$, and $\max_{j \in W} \tilde{c}_j(\lambda, C_{2k}) \leq \max_{j \in W} b'_j$, and $\tilde{c}(\lambda, C_{2k}) \leq \sum_j b'_j$.)

Finally, by the definition of the capped cost, the bound on $\max_j \tilde{c}_j(\lambda, C_{2k})$ in Observation 3.2 implies that $\tilde{c}(\lambda, C_{2k}) = (1 - 2k/n)c(C_{2k})$. This and the bound on $\tilde{c}(\lambda, C_{2k})$ in Observation 3.2 imply $c(C_{2k}) \leq \lambda$, so that the loop terminates after iteration $2k$. This proves Theorem 3.2. ∎

4. Fast algorithm. The running time of the algorithm in Corollary 3.3 is dominated by the time to compute the optimal LP solution. But the algorithm uses only a single parameter of that solution, namely its cost λ^*. This section builds on that to give our main result—a faster algorithm. It replaces the first phase by an algorithm that, instead of solving the LP, somehow computes a pair (λ, C) such that $\lambda \leq \lambda^*$ and $\tilde{c}(\lambda, C) < 1$:

THEOREM 4.1. *Given just the instance c, in $O(k\,m \log(n/k) \log m)$ time one can compute a $\lambda \leq \lambda^*$ and a set $C \subseteq U$ of size at most T such that $\tilde{c}(\lambda, C) < 1$.*

With Theorem 3.2, this will give the following result:

COROLLARY 4.2. *K-Median admits an α_{nk}-size-approximation algorithm that runs in $O(k\,m \log(n/k) \log m)$ time.* ∎

Proof of Theorem 4.1. The algorithm to compute (λ, C) is greedy$'(c)$ in Figure 4.1. ∎

Correctness. Consider executing greedy$'(c)$. Let C_t denote C at the end of iteration t (and $C_0 = \emptyset$). We'll show that the algorithm maintains the invariant

$$\lambda_t \leq \lambda^* \text{ and } \tilde{c}(\lambda_t, C_t) \leq p^t n/(2k+1) + (1 - p^t)(1 - 2k/n) \tag{4.1}$$

Note the similarity to (3.1) in the proof of Theorem 3.1.

Invariant (4.1) is true initially because $\lambda_0 = 0$ and $\tilde{c}(\lambda_0, \emptyset) = n/(2k+1)$.

Suppose the invariant holds just before a given iteration t. That is,

$$\lambda_{t-1} \leq \lambda^* \text{ and } \tilde{c}(\lambda_{t-1}, C_{t-1}) \leq p^{t-1}n/(2k+1) + (1-p^{t-1})(1-2k/n).$$

First we argue that $\lambda_t \leq \lambda^*$. In the case that $\lambda_t = \lambda_{t-1}$, this follows from $\lambda_{t-1} \leq \lambda^*$. Otherwise, consider any λ in the half-open interval $[\lambda_{t-1}, \lambda_t)$. The algorithm's choice of λ_t (using that $\tilde{c}(\lambda, C)$ is non-increasing with λ) ensures

$$\begin{aligned}\min_{i \in U} \tilde{c}(\lambda, C_{t-1} \cup \{i\}) &> (1-1/k)\tilde{c}(\lambda_{t-1}, C_{t-1}) + (1/k)(1-2k/n) \\ &\geq (1-1/k)\tilde{c}(\lambda, C_{t-1}) + (1/k)(1-2k/n),\end{aligned}$$

which, with Lemma 2.3, implies that $\lambda^*/\lambda > 1$. So $\lambda < \lambda^*$ for all λ in the non-empty interval $[\lambda_{t-1}, \lambda_t)$. The desired bound $\lambda^* \geq \lambda_t$ follows.

To finish showing that (4.1) is maintained we bound $\tilde{c}(\lambda_t, C_t)$. Recall that $p = 1 - 1/k$. The choices of i' and τ, and the invariant at time $t-1$, give

$$\begin{aligned}\tilde{c}(\lambda, C_t \cup \{i'\}) &\leq p\,\tilde{c}(\lambda_{t-1}, C_{t-1}) + (1-p)(1-2k/n) \\ &\leq p\big[p^{t-1}n/(2k+1) + (1-p^{t-1})(1-2k/n)\big] + (1-p)(1-2k/n) \\ &= p^t n/(2k+1) + (1-p^t)(1-2k/n).\end{aligned}$$

Hence invariant (4.1) is maintained. Now suppose for contradiction that there are more than T iterations. After iteration T, the invariant implies that $\lambda_T \leq \lambda^*$, and by the choice of $p = 1 - 1/k < e^{-1/k}$ and $T \geq k\ln(n^2/(2k(2k+1))$ we have

$$\tilde{c}(\lambda_T, C_T) \leq e^{-T/k}\frac{n}{2k+1} + 1 - \frac{2k}{n} < \frac{2k(2k+1)}{n^2} \times \frac{n}{2k+1} + 1 - \frac{2k}{n} = 1,$$

so the loop terminates after iteration T, contradicting that there are more than T iterations. Hence the pair (λ_t, C_t) returned by the algorithm is as claimed in Theorem 4.1.

Run time. The loop makes at most $T = O(k\log(n/k))$ iterations. To show the claimed time bound, we show that each loop iteration can be implemented using $O(\log m)$ iterations of binary search, each of which takes $O(m)$ time.

For each center/customer pair $(i,j) \in W \times U$ define $\beta_{ij} = (2k+1)(1-2k/n)c_{ij}$. Define $(\beta_1, \beta_2, \ldots, \beta_N)$ to be the set $\{\beta_{ij} : i \in U, j \in W\} \cup \{\beta_0, \beta_\infty\}$ of *breakpoints*, in increasing order. Note that $\beta_1 = 0$, $\beta_N = \infty$, and $N \leq m+2$.

In a given loop iteration $t \leq T$, compute λ_t as follows.

Given any λ, we can query the condition "$\lambda_t \leq \lambda$" in $O(m)$ time, using that $\lambda_t \leq \lambda$ if and only if $\min_{i \in U} \tilde{c}(\lambda, C \cup \{i\}) \leq \tau$. Using this, first check whether $\lambda_t \leq \lambda_{t-1}$. If it holds we have found λ_t (as $\lambda_t = \lambda_{t-1}$), so assume $\lambda_t > \lambda_{t-1}$. Use $O(\log m)$ iterations of binary search (checking $\lambda_t \leq \beta_\ell$ for some $\ell \in [N-1]$ in each of these iterations) to find $\ell \in [N-1]$ such that that $\beta_\ell < \lambda_t \leq \beta_{\ell+1}$.

The capped cost $\tilde{c}(\lambda, C \cup \{i\})$ implicitly defines a "partial" assignment that assigns each customer j to its closest center $g_j = \arg\min_{g \in C \cup \{i\}} c_{gj}$ in $C \cup \{i\}$ if $c_{g_j,j}(1 - 2k/n)/\lambda < 1/(2k+1)$, and otherwise leaves j unassigned. The open interval $(\beta_\ell, \beta_{\ell+1})$ contains no breakpoints, so this assignment is the same for all λ in this interval. Hence, letting $u(i)$ be the number of customers not assigned by the assignment, and letting $c'(i)$ denote the total cost of just the assigned customers, for any λ in $[\beta_\ell, \beta_{\ell+1}]$ the

sample(c, x^*, y^*) — (x^*, y^*) *is optimal solution to the LP*

1. assign $a_j \leftarrow$ none for $j \in W$ — *initialize partial assignment* a
2. do the following steps T times:
3. choose center $i \in U$ randomly from distribution x^*/k
4. for each customer $j \in W$: with probability y^*_{ij}/x^*_i, reassign $a_j \leftarrow i$
5. return a

FIG. 5.1. *Rounding to a partial assignment by sampling.*

capped cost $\tilde{c}(\lambda, C \cup \{i\})$ equals $c'(i)(1 - 2k/n)/\lambda + u(i)/(2k+1)$. Hence,

$$\begin{aligned}
\lambda_t &= \min\big\{\lambda \ge \lambda_{t-1} : \min_{i\in U} \tilde{c}(\lambda, C \cup \{i\}) \le \tau\big\} \\
&= \min\big\{\lambda \ge \lambda_{t-1} : \min_{i\in U} c'(i)(1 - 2k/n)/\lambda + u(i)/(2k+1) \le \tau\big\} \\
&= \min\big\{\lambda \ge \lambda_{t-1} : \min_{i\in U} c'(i)(1 - 2k/n)/(\tau - u(i)/(2k+1)) \le \lambda\big\} \\
&= \max\big(\lambda_{t-1},\, \min_{i\in U} c'(i)(1 - 2k/n)/(\tau - u(i)/(2k+1))\big).
\end{aligned}$$

After computing $u(i)$ and $c'(i)$ for all $i \in U$ in $O(m)$ time (total), the algorithm computes λ_t (as the right-hand side above) in $O(m)$ time. (In the case $\min_{i\in U} u(i) = n$, this gives $\lambda_t = \infty$.) Given λ_t, it then computes the center i' to add in $O(m)$ time. This proves Theorem 4.1. ∎

5. The capped cost is a pessimistic estimator. This section gives some intuition for the somewhat mysterious capped cost $\tilde{c}(\lambda, C)$. Briefly, it is a pessimistic estimator from the analysis of a natural random experiment.

DEFINITION 5.1. *A* partial assignment *is a vector* $a \in (U \cup \{\text{none}\})^W$*, with the interpretation that* a_j *is the center assigned to customer* $j \in W$*, or* $a_j =$ none *if* j *has no assigned center. Let the* cost *of* a *be* $c(a) = \sum_{j:a_j \ne \text{none}} c_{a_j,j}$*. Let* $u(a) = |\{j \in W : a_j = \text{none}\}|$ *denote the number of unassigned customers.*

Consider the partial rounding scheme in Figure 5.1. It takes as input the instance c and the optimal solution (x^*, y^*) to the LP. It rounds the fractional solution to a partial assignment by sampling $T = \lceil k \ln(n^2/(2k(2k+1)))\rceil$ times (with replacement) from the distribution x^*/k, and, with each sampled center i, assigning (or reassigning) each customer $j \in W$ to i with probability $y^*_j/x^*_i \le 1$. In this way the rounding scheme maintains a partial assignment a. Note that a can assign a customer to a center that is not its closest open center. It returns the partial assignment resulting from T such iterations.

Let random variable A be the partial assignment returned by sample(x^*, y^*).

LEMMA 5.2. $\mathbb{E}[c(A)] \le \lambda^*$ *and* $\mathbb{E}[u(A)] < 2k(2k+1)/n$.

Proof sketch.. As shown in [9], for any center $i \in U$ and customer $j \in W$ the probability that any single iteration assigns customer j to some center is $1/k$. Also, given that j is assigned to some center (in any iteration), the expected cost of the assignment is $\sum_i y^*_{ij} c_{ij}$, so

$$\begin{aligned}
\mathbb{E}[c(A)] &= \textstyle\sum_{ij} \Pr[a_j \ne \text{none}] \times \Pr[a_j = i \,|\, a_j \ne \text{none}]\, c_{ij} \\
&\le \textstyle\sum_{ij} \Pr[a_j = i \,|\, a_j \ne \text{none}]\, c_{ij} \\
&= \textstyle\sum_{ij} y^*_{ij}\, c_{ij} \;=\; c \cdot y^* \;=\; \lambda^*.
\end{aligned}$$

Meanwhile, the probability that a given customer $j \in W$ is never assigned is $(1-1/k)^T < \exp^{-T/k} \le 2k(2k+1)/n^2$. The desired bound $\mathbb{E}[u(A)] < 2k(2k+1)/n$ follows by linearity of expectation. ∎

LEMMA 5.3. *With positive probability, A has cost $c(A) \le \lambda^*/(1-2k/n)$ and $u(A) \le 2k$.*

Proof sketch.. Assume $0 < \lambda^* < \infty$. (The other cases are easy to verify.)

$$\begin{aligned}
&\Pr\left[c(A) \ge \lambda^*/(1-2k/n) \text{ or } u(A) \ge 2k+1\right] \\
&\quad \le \Pr[c(A) \ge \lambda^*/(1-2k/n)] + \Pr[u(A) \ge 2k+1] && \text{(naive union bound)} \\
&\quad \le \mathbb{E}[c(A)]\frac{1-2k/n}{\lambda^*} + \mathbb{E}[u(A)]\frac{1}{2k+1} && \text{(Markov bound)} \qquad (5.1) \\
&\quad < 1 - 2k/n + 2k/n = 1 && \text{(Lemma 5.2).}
\end{aligned}$$

∎

The proof of the lemma bounds the probability of the two "bad" events by the expectation of a pessimistic estimator, then shows that the expectation of that pessimistic estimator is less than 1. Instead of using Lemma 5.3 directly, we work directly with the pessimistic estimator, which is the capped cost $\tilde{c}(\lambda, A)$ for $\lambda = \lambda^*$.

In fact, the greedy algorithm in the first phase of the slow algorithm can be obtained by applying the method of conditional probabilities to the random-sampling rounding scheme (following the approach initiated in [8]) to find an outcome with $\tilde{c}(\lambda^*, A) < 1$.

6. Conclusion. To conclude we remark on the LP dual solutions implicitly generated by the algorithm, sketch how the algorithms compare to those in previous works, and give some open problems.

Implicit LP dual solutions. As expected, the algorithms presented here implicitly define solutions to the dual of the standard k-Median LP relaxation (Figure 1.1). The proven approximation ratios hold with respect to the dual solution cost.

Briefly, rewriting the bound in Lemma 2.3, it is

$$\lambda^* \ge \frac{\lambda}{1-2k/n}\big(\tilde{c}(\lambda, C) - k \max_{i \in U}[\tilde{c}(\lambda, C) - \tilde{c}(\lambda, C \cup \{i\})]\big).$$

This lower bound is equivalent to weak duality for the feasible dual solution (δ, π, μ) defined by

$$\begin{aligned}
\delta_j &= \tfrac{\lambda}{1-2k/n}\tilde{c}_j(\lambda, C) = \min\left(\tfrac{\lambda}{(1-2k/n)(2k+1)}, \min_{i \in C} c_{ij}\right), \\
\pi_{ij} &= \max(0, \delta_j - c_{ij}), \text{ and } \mu = \max_{i \in U} \textstyle\sum_j \pi_{ij}.
\end{aligned}$$

Each iteration of the first phase, the current pair (λ, C) defines a feasible dual solution whose cost is a lower bound on λ^*. The cost of the final primal solution is at most the maximum cost of any of these dual solutions, which is in turn at most λ^*. This can be shown by mechanically recasting the relevant invariants in terms of the dual solutions.

Comparison to previous works. The bicriteria-approximation algorithm for k-Median in [9, §6] takes as input an instance c, an $\epsilon > 0$, and any upper bound λ on the optimal fractional cost λ^*. It returns a solution of cost at most $(1+\epsilon)\lambda$ of size at most $1 + k\ln(n + n/\epsilon)$. (Note that the size is $\Omega(k \ln n)$ regardless of ϵ.) That algorithm is derived by derandomizing a natural random-sampling rounding scheme.

The first $O(\log n)$-size approximation algorithm, in [1, Theorem 5], requires as input the instance *and* the optimal cost λ^*. It calls the algorithm from [9] with $\epsilon = 1/n$ and $\lambda = \lambda^*$ to obtain a set C of centers, then returns $C \cup \{i\}$, where i is chosen to minimize the cost of $C \cup \{i\}$. Assuming without loss of generality that every customer has a center with assignment cost zero, this reduces the cost by at least a factor of $1 - 1/n$, reducing the cost below λ^*.

The first phase of our first algorithm can be obtained by derandomizing the rounding scheme from [9], but with respect to a different analysis, so ours makes fewer iterations and minimizes a different function. Then, instead of adding just one center to bring the cost down, our polishing step (which is the key to obtaining the stronger approximation ratio) adds $2k$ centers. Finally, all the previous size-approximation algorithms require solving the LP. Our faster algorithm avoids this as described in Section 4.

Open problems. Is there a polynomial-time algorithm with size-approximation ratio $\alpha \ln(n/k) + o(\log(n/k))$ for some constant $\alpha < 2$?

In the more general *weighted* k-Median problem, each center i is given a weight $w_i \geq 0$, and the set of centers must have *total weight* at most k, rather than *size* at most k. Does weighted k-Median have a polynomial-time $O(\log(n/k))$-size approximation algorithm? The result in [9] extends to this problem.

For the closely related Facilities Location problem, the best polynomial-time approximation algorithm returns a solution of cost at most $c \cdot y^* + H_\Delta f \cdot x^*$, where (x^*, y^*) is an optimum solution to the standard LP relaxation for Facilities Location. That is, it achieves ratio H_Δ with respect to the opening costs, and ratio 1 with respect to the assignment costs. This algorithm is relatively slow because it must solve the LP to obtain (x^*, y^*). Is there a faster greedy algorithm with the same performance guarantee?

REFERENCES

[1] M. Chrobak, C. Kenyon, J. Noga, and N. E. Young, *Incremental medians via online bidding*, Algorithmica, 50 (2008), pp. 455–478, https://doi.org/10.1007/s00453-007-9005-x.

[2] V. Chvatal, *A greedy heuristic for the set-covering problem*, Mathematics of Operations Research, 4 (1979), pp. 233–235, https://doi.org/10.1287/moor.4.3.233.

[3] U. Feige, *A threshold of* $\ln n$ *for approximating Set Cover*, Journal of the ACM, 45 (1998), pp. 634–652, https://doi.org/10.1145/285055.285059.

[4] D. S. Johnson, *Approximation algorithms for combinatorial problems*, Journal of Computer and System Sciences, 9 (1974), pp. 256–278, https://doi.org/10.1016/S0022-0000(74)80044-9.

[5] J.-H. Lin and J. S. Vitter, *ε-approximations with minimum packing constraint violation (extended abstract)*, in Proceedings of the twenty-fourth annual ACM Symposium on Theory of Computing, STOC '92, New York, NY, USA, July 1992, Association for Computing Machinery, pp. 771–782, https://doi.org/10.1145/129712.129787.

[6] L. Lovász, *On the ratio of optimal integral and fractional covers*, Discrete Mathematics, 13 (1975), pp. 383–390, https://doi.org/10.1016/0012-365X(75)90058-8.

[7] P. Slavík, *A tight analysis of the greedy algorithm for Set Cover*, Journal of Algorithms, 25 (1997), pp. 237–254, https://doi.org/10.1006/jagm.1997.0887.

[8] N. E. Young, *Randomized rounding without solving the linear program*, in Proceedings of the sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1995, Philadelphia, PA, USA, 1995, Society for Industrial and Applied Mathematics, pp. 170–178, https://dl.acm.org/doi/10.5555/313651.313689.

[9] N. E. Young, *K-medians, Facility Location, and the Chernoff-Wald bound*, in Proceedings of the eleventh annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2000, Philadelphia, PA, USA, 2000, Society for Industrial and Applied Mathematics, pp. 86–95, https://dl.acm.org/doi/10.5555/338219.338239.

Appendix A. Missing proofs.

Proof of Lemma 2.1.

$$\begin{aligned}
\alpha_{kn} = T/k + 2 &\le \ln(n^2/(2k(2k+1))) + 1/k + 2 && \text{(by defn of } T\text{)} \\
&= 2\ln(n/k) + 2 - 2\ln 2 + 1/k + \ln(1 - 1/(2k+1)) \\
&\le 2\ln(n/k) + 2 - 2\ln 2 + 1/k - 1/(2k+1) && (\ln(1+z) \le z) \\
&= 2\ln(n/k) + 2 - 2\ln 2 + 1/(2k) + 1/(2k(2k+1)) && \square
\end{aligned}$$

Proof of Lemma 2.3. Consider the following random experiment, from [9]. Let (x^*, y^*) be an optimal solution (of cost λ^*) to the k-Median LP relaxation (Figure 1.1). Choose a center $i' \in U$ randomly from the distribution x^*/k, then, for each customer $j \in W$ independently, reassign j to i' (in place of whatever current assignment it has) with probability $y^*_{i'j}/x_i$.

As observed in [9], for any customer $j \in W$, the probability that j is reassigned is $1/k$, and the expected cost of j's new assignment, given that it is reassigned, is $\sum_{i=1}^{n} y^*_{ij} c_{ij}$, so

$$\begin{aligned}
\min_{i \in U} \tilde{c}(\lambda, C \cup \{i\}) \le \mathbb{E}[\tilde{c}(\lambda, C \cup \{i'\})] &= \sum_{j \in W} \mathbb{E}[\tilde{c}_j(\lambda, C \cup \{i\})] \\
&\le \sum_{j \in W} (1 - 1/k)\tilde{c}_j(\lambda, C) + (1/k)\textstyle\sum_i y^*_{ij}(1 - 2k/n)c_{ij}/\lambda \\
&= (1 - 1/k)\tilde{c}(\lambda, C) + (1/k)(1 - 2k/n)\lambda^*/\lambda. \qquad \square
\end{aligned}$$