Universal Algorithms for Clustering Problems

ARUN GANESH, UC Berkeley, USA
BRUCE M. MAGGS, Duke University and Emerald Innovations, USA
DEBMALYA PANIGRAHI, Duke University, USA

This article presents *universal* algorithms for clustering problems, including the widely studied k-median, k-means, and k-center objectives. The input is a metric space containing all *potential* client locations. The algorithm must select k cluster centers such that they are a good solution for *any* subset of clients that actually realize. Specifically, we aim for low *regret*, defined as the maximum over all subsets of the difference between the cost of the algorithm's solution and that of an optimal solution. A universal algorithm's solution $\textsc{sol}$ for a clustering problem is said to be an (α, β)-approximation if for all subsets of clients C', it satisfies $\textsc{sol}(C') \leq \alpha \cdot \textsc{opt}(C') + \beta \cdot \textsc{mr}$, where $\textsc{opt}(C')$ is the cost of the optimal solution for clients C' and $\textsc{mr}$ is the minimum regret achievable by any solution.

Our main results are universal algorithms for the standard clustering objectives of k-median, k-means, and k-center that achieve $(O(1), O(1))$-approximations. These results are obtained via a novel framework for universal algorithms using linear programming (LP) relaxations. These results generalize to other ℓ_p-objectives and the setting where some subset of the clients are *fixed*. We also give hardness results showing that (α, β)-approximation is NP-hard if α or β is at most a certain constant, even for the widely studied special case of Euclidean metric spaces. This shows that in some sense, $(O(1), O(1))$-approximation is the strongest type of guarantee obtainable for universal clustering.

CCS Concepts: • **Theory of computation → Facility location and clustering;**

Additional Key Words and Phrases: Universal algorithms, clustering

ACM Reference format:
Arun Ganesh, Bruce M. Maggs, and Debmalya Panigrahi. 2023. Universal Algorithms for Clustering Problems. *ACM Trans. Algor.* 19, 2, Article 15 (March 2023), 46 pages.
https://doi.org/10.1145/3572840

Arun Ganesh was supported in part by NSF Award CCF-1535989. Bruce M. Maggs was supported in part by NSF Award CCF-1535972. Debmalya Panigrahi was supported in part by NSF grants CCF-1535972, CCF-1955703, an NSF CAREER Award CCF-1750140, and the Indo-US Virtual Networked Joint Center on Algorithms under Uncertainty.
Authors' addresses: A. Ganesh, UC Berkeley, Soda Hall, Berkeley, California, USA, 94709; email: arunganesh@berkeley.edu; B. M. Maggs, Duke University and Emerald Innovations, 308 Research Drive, Durham, North Carolina, USA, 27710; email: bmm@cs.duke.edu; D. Panigrahi, Duke University, 308 Research Drive, Durham, North Carolina, USA, 27710; email: debmalya@cs.duke.edu.

1549-6325/2023/03-ART15 $15.00
https://doi.org/10.1145/3572840

1 INTRODUCTION

In *universal*[1] approximation (e.g., References [9, 10, 12, 22, 26, 31, 36, 49, 50]), the algorithm is presented with a set of *potential* input points and must produce a solution. After seeing the solution, an adversary selects some subset of the points as the actual *realization* of the input, and the cost of the solution is based on this realization. The goal of a universal algorithm is to obtain a solution that is near-optimal for *every* possible input realization. For example, suppose that a network-based-service provider can afford to deploy servers at k locations around the world and hopes to minimize latency between clients and servers. The service provider does not know in advance which clients will request service but knows where clients are located. A universal solution provides guarantees on the quality of the solution regardless of which clients ultimately request service. As another example, suppose that a program committee chair wishes to invite k people to serve on the committee. The chair knows the areas of expertise of each person who is qualified to serve. Based on past iterations of the conference, the chair also knows about many possible topics that might be addressed by submissions. The chair could use a universal algorithm to select a committee that will cover the topics well, regardless of the topics of the papers that are submitted. The situation also arises in targeting advertising campaigns to client demographics. Suppose a campaign can spend for k advertisements, each targeted to a specific client type. While the entire set of client types that are potentially interested in a new product is known, the exact subset of clients that will watch the ads, or eventually purchase the product, is unknown to the advertiser. How does the advertiser target her k advertisements to address the interests of any realized subset of clients?

Motivated by these sorts of applications, this article presents the first universal algorithms for clustering problems, including the classic k-median, k-means, and k-center problems. The input to these algorithms is a metric space containing all locations of *clients* and *cluster centers*. The algorithm must select k cluster centers such that this is a good solution for *any* subset of clients that actually realize.

It is tempting to imagine that, in general, for some large-enough value of α, one can find a solution SOL such that for all realizations (i.e., subsets of clients) C', $\text{SOL}(C') \leq \alpha \cdot \text{OPT}(C')$, where $\text{SOL}(C')$ denotes SOL's cost in realization C' and $\text{OPT}(C')$ denotes the optimal cost in realization C'. But this turns out to be impossible for many problems, including the clustering problems we study, and indeed this difficulty may have limited the study of universal algorithms. For example, suppose that the input for the k-median problem is a uniform metric on $k+1$ points, each with a cluster center and client. In this case, for any solution SOL with k cluster centers, there is some realization C' consisting of a single client that is not co-located with any of the k cluster centers in SOL. Then, $\text{SOL}(C') > 0$ but $\text{OPT}(C') = 0$. Since it is not possible to provide a strict multiplicative approximation guarantee for every realization, we could instead consider an additive approximation. That is, we could instead seek to minimize the *regret*, defined as the maximum difference between the cost of the algorithm's solution and the optimal cost across all realizations. Informally, the regret is the additional cost incurred due to not knowing the realization ahead of time. The solution that minimizes regret is called the *minimum regret solution*, or MRS for short, and its regret is termed *minimum regret* or MR. More formally, $\text{MR} = \min_{\text{SOL}} \max_{C'} [\text{SOL}(C') - \text{OPT}(C')]$. We now seek a solution SOL that achieves, for all input realizations C', $\text{SOL}(C') - \text{OPT}(C') \leq \text{MR}$, i.e., $\text{SOL}(C') \leq \text{OPT}(C') + \text{MR}$. But obtaining such a solution turns out to be **NP**-hard for many problems, and furthermore obtaining a solution that achieves approximately minimum regret (that is, it has

[1] In the context of clustering, universal facility location sometimes refers to facility location where facility costs scale with the number of clients assigned to them. This problem is unrelated to the notion of universal algorithms studied in this article.

regret at most $c \cdot \text{MR}$ for some c) is also **NP**-hard in general (see Section 8 for more details). In turn, one has to settle for approximation on both OPT and MR. That is, we settle for seeking SOL such that $\text{SOL}(C') \leq \alpha \cdot \text{OPT}(C') + \beta \cdot \text{MR}$ for all C'. An algorithm generating such a solution is then called an (α, β)-approximate universal algorithm for the problem. Note that in the aforementioned example with $k+1$ points, any solution must pay MR (the distance between any two points) in some realization where $\text{OPT}(C') = 0$ and only one client appears (in which case paying MR might sound avoidable or undesirable). This example demonstrates that stricter notions of regret and approximation than (α, β)-approximation are infeasible in general, suggesting that (α, β)-approximation is the least relaxed guarantee possible for universal clustering.

1.1 Problem Definitions and Results

We are now ready to formally define our problems and state our results. In all the clustering problems that we consider in this article, the input is a metric space on all the potential client locations C and cluster centers F.[2,3] Let c_{ij} denote the metric distance between points i and j. The solution produced by the algorithm comprises k cluster centers in F; let us denote this set by SOL. Now, suppose a subset of clients $C' \subseteq C$ realizes in the actual input. Then, the cost of each client $j \in C'$ is given as the distance from the client to its closest cluster center, i.e., $\text{COST}(j, \text{SOL}) = \min_{i \in \text{SOL}} c_{ij}$. The clustering problems differ in how these costs are combined into the overall minimization objective. The respective objectives are as follows:

- *k-median* (e.g., Reference [6, 13, 17, 34, 44]): *sum* of client costs, i.e., $\text{SOL}(C') = \sum_{j \in C'} \text{COST}(j, \text{SOL})$.
- *k-center* (e.g., Reference [21, 32, 33, 40, 47]): *maximum* client cost, i.e., $\text{SOL}(C') = \max_{j \in C'} \text{COST}(j, \text{SOL})$.
- *k-means* (e.g., Reference [1, 30, 37, 43, 45]): ℓ_2*-norm* of client costs, i.e., $\text{SOL}(C') = \sqrt{\sum_{j \in C'} \text{COST}(j, \text{SOL})^2}$.

We also consider ℓ_p**-clustering** (e.g., Reference [30]), which generalizes all these individual clustering objectives. In ℓ_p-clustering, the objective is the ℓ_p-norm of the client costs for a given value $p \geq 1$, i.e.,

$$\text{SOL}(C') = \left(\sum_{j \in C'} \text{COST}(j, \text{SOL})^p \right)^{1/p}.$$

Note that k-median and k-means are special cases of ℓ_p-clustering for $p = 1$ and $p = 2$, respectively. k-center can also be defined in the ℓ_p-clustering framework as the limit of the objective for $p \to \infty$; moreover, it is well known that ℓ_p-norms only differ by constants for $p > \log n$ (see Appendix A), thereby allowing the k-center objective to be approximated within a constant by ℓ_p-clustering for $p = \log n$.

Our main result is to obtain $(O(1), O(1))$-approximate universal algorithms for k-median, k-center, and k-means. We also generalize these results to the ℓ_p-clustering problem.

[2]We only consider finite C in this article. If C is infinite (e.g., $C = \mathbb{R}^d$), then the minimum regret will usually also be infinite. If one restricts to realizations where, say, at most m clients appear, then it suffices to consider realizations that place m clients at one of finitely many points, letting us reduce to universal k-center with finite C.

[3]The special case where $F = C$ has also been studied in the clustering literature, e.g., in References [17, 32], although the more common setting (as in our work) is to not make this assumption. Of course, all results (including ours) without this assumption also apply to the special case. If $F = C$, then the constants in our bounds improve, but the results are qualitatively the same. We note that some sources refer to the k-center problem when $F \neq C$ as the k-supplier problem instead, and use k-center to refer exclusively to the case where $F = C$.

Theorem 1.1. *There are $(O(1), O(1))$-approximate universal algorithms for the k-median, k-means, and k-center problems. More generally, there are $(O(p), O(p^2))$-approximate universal algorithms for ℓ_p-clustering problems for any $p \geq 1$.*

Remark. The bound for k-means is by setting $p = 2$ in ℓ_p-clustering. For k-median and k-center, we use separate algorithms to obtain improved bounds than those provided by the ℓ_p-clustering result. This is particularly noteworthy for k-center where ℓ_p-clustering only gives poly-logarithmic approximation.

Universal Clustering with Fixed Clients. We also consider a more general setting where some of the clients are *fixed*, i.e., are there in any realization, but the remaining clients may or may not realize as in the previous case. (Of course, if no client is fixed, then we get back the previous setting as a special case.) This more general model is inspired by settings where a set of clients is already present but the remaining clients are mere predictions. This surprisingly creates new technical challenges, that we overcome to get the following:

Theorem 1.2. *There are $(O(1), O(1))$-approximate universal algorithms for the k-median, k-means, and k-center problems with fixed clients. More generally, there are $(O(p^2), O(p^2))$-approximate universal algorithms for ℓ_p-clustering problems, for any $p \geq 1$.*

Hardness Results. Next, we study the limits of approximation for universal clustering. In particular, we show that the universal clustering problems for all the objectives considered in this article are **NP**-hard in a rather strong sense. Specifically, we show that both α and β are separately bounded away from 1, *irrespective of the value of the other parameter*, showing the necessity of both α and β in our approximation bounds. Similar lower bounds continue to hold for universal clustering in Euclidean metrics, even when PTASes are known in the offline (non-universal) setting [1, 5, 41, 43, 47].

Theorem 1.3. *In universal ℓ_p-clustering for any $p \geq 1$, obtaining $\alpha < 3$ or $\beta < 2$ is **NP**-hard. Even for Euclidean metrics, obtaining $\alpha < 1.8$ or $\beta \leq 1$ is **NP**-hard. The lower bounds on α (respectively, β) are independent of the value of β (respectively, α).*

Interestingly, our lower bounds rely on realizations where sometimes as few as one client appears. This suggests that, e.g., redefining regret to be some function of the number of clients that appear (rather than just their cost) cannot subvert these lower bounds.

1.2 Techniques

Before discussing our techniques, we discuss why standard approximations for clustering problems are insufficient. It is known that the *optimal* solution for the realization that includes all clients gives a $(1, 2)$-approximation for universal k-median (this is a corollary of a more general result in Reference [38]; we do not know if their analysis can be extended to, e.g., k-means), giving universal algorithms for "easy" cases of k-median such as tree metrics. But the clustering problems we consider in this article are **NP**-hard in general; so, the best we can hope for in polynomial time is to obtain optimal *fractional* solutions, or *approximate* integer solutions. Unfortunately, the proof of Reference [38] does not generalize to *any* regret guarantee for the optimal *fractional* solution. Furthermore, for all problems considered in this article, even $(1+\epsilon)$-approximate (integer) solutions for the "all clients" instance are not guaranteed to be (α, β)-approximations for any finite α, β (see the example in Appendix B). These observations fundamentally distinguish universal approximations for **NP**-hard problems like the clustering problems in this article from those in **P**, and require us to develop new techniques for universal approximations.

In this article, we develop a general framework for universal approximation based on linear programming (LP) relaxations that forms the basis of our results on k-median, k-means, and k-center (Theorem 1.1) as well as the extension to universal clustering with fixed clients (Theorem 1.2).

The first step in our framework is to write an LP relaxation of the regret minimization problem. In this formulation, we introduce a new regret variable that we seek to minimize and is constrained to be at least the difference between the (fractional) solution obtained by the LP and the optimal integer solution *for every realizable instance*. Abstractly, if the LP relaxation of the optimization problem is given by $\min\{\mathbf{c}\cdot\mathbf{x} : \mathbf{x}\in P\}$, then the new *regret minimization LP*[4] is given by

$$\min\{\mathbf{r} : \mathbf{x}\in P; \mathbf{c}(I)\cdot\mathbf{x}\leq \textsc{opt}(I)+\mathbf{r},\ \forall I\}.$$

Here, I ranges over all realizable instances of the problem. Hence, the LP is exponential in size, and we need to invoke the ellipsoid method via a separation oracle to obtain an optimal fractional solution. We first note that the constraints $\mathbf{x}\in P$ can be handled using a separation oracle for the optimization problem itself. So, our focus is on designing a separation oracle for the new set of constraints $\mathbf{c}(I)\cdot\mathbf{x}\leq \textsc{opt}(I)+\mathbf{r},\ \forall I$. This amounts to determining the regret of a fixed solution given by $\mathbf{x}$, which unfortunately, is **NP**-hard for our clustering problems. So, we settle for designing an approximate separation oracle, i.e., approximating the regret of a given solution. For k-median, we reduce this to a submodular maximization problem subject to a cardinality constraint, which can then be (approximately) solved via standard greedy algorithms. For k-means, and more generally ℓ_p-clustering, the situation is more complex. Similarly to k-median, maximizing regret for a fixed solution can be reduced to a set of submodular maximization problems, but deriving the functional value for a given set now requires solving the **NP**-hard knapsack problem. We overcome this difficulty by showing that we can use *fractional* knapsack solutions as surrogates for the optimal integer knapsack solution in this reduction, thereby restoring polynomial running time of the submodular maximization oracle. Finally, in the presence of fixed clients, we need to run the submodular maximization algorithm over a set of combinatorial objects called *independence systems*. In this case, the resulting separation oracle only gives a bicriteria guarantee, i.e., the solutions it considers feasible are only guaranteed to satisfy $\forall I : \mathbf{c}(I)\cdot\mathbf{x}\leq \alpha\cdot\textsc{opt}(I)+\beta\cdot\mathbf{r}$ for constants α and β. Note that the bi-criteria guarantee suffices for our purposes since these constants are absorbed in the overall approximation bounds.

The next step in our framework is to round these fractional solutions to integer solutions for the regret minimization LP. Typically, in clustering problems such as k-median, LP rounding algorithms give *average* guarantees, i.e., although the overall objective in the integer solution is bounded against that of the fractional solution, individual connection costs of clients are not (deterministically) preserved in the rounding. But average guarantees are too weak for our purpose: in a realized instance, an adversary may only select the clients whose connection costs increase by a large factor in the rounding thereby causing a large regret. Ideally, we would like to ensure that the connection cost of *every* individual client is preserved up to a constant in the rounding. However, this may be impossible in general, i.e., no integer solution might satisfy this requirement. Consider a uniform metric over $k+1$ points. One fractional solution is to make $\frac{k}{k+1}$ fraction of each point a cluster center. Then, each client has connection cost $\frac{1}{k+1}$ in the fractional solution since it needs to connect $\frac{1}{k+1}$ fraction to a remote point. However, in any integer solution, since there are only k cluster centers but $k+1$ points overall, there is one client that has connection cost of 1, which is $k+1$ times its fractional connection cost.

[4]For problems like k-means with non-linear objectives, the constraint $\mathbf{c}(I)\cdot\mathbf{x}\leq \textsc{opt}(I)+\mathbf{r}$ cannot be replaced with a constraint that is simultaneously linear in $\mathbf{x}$, $\mathbf{r}$. However, for a fixed value of $\mathbf{r}$, the corresponding non-linear constraints still give a convex feasible region, and so the techniques we discuss in this section can still be used.

To overcome this difficulty, we allow for a uniform *additive* increase in the connection cost of every client. We show that such a rounding also preserves the regret guarantee of our fractional solution within constant factors. The clustering problem we now solve has a modified objective: for every client, the distance to the closest cluster center is now discounted by the additive allowance, with the caveat that the connection cost is 0 if this difference is negative. This variant is a generalization of a problem appearing in Reference [25], and we call it clustering *with discounts* (e.g., for k-median, we call this problem *k-median with discounts*.) Our main tool in the rounding then becomes an approximation algorithm to clustering problems with discounts. For k-median, we use a Lagrangian relaxation of this problem to the classic facility location problem to design such an approximation. For k-means and ℓ_p-clustering, the same general concept applies, but we need an additional ingredient called a *virtual* solution that acts as a surrogate between the regret of the (rounded) integer solution and that of the fractional solution obtained above. For k-center, we give a purely combinatorial (greedy) algorithm.

1.3 Related Work

For all previous universal algorithms, the approximation factor corresponds to our parameter α, i.e., these algorithms are $(\alpha, 0)$-approximate. The notion of regret was not considered. As we have explained, however, it is not possible to obtain such results for universal clustering. Furthermore, it may be possible to trade off some of the large values of α in the results below, e.g., $\Omega(\sqrt{n})$ for set cover, by allowing $\beta > 0$.

Universal algorithms have been of large interest in part because of their applications as online algorithms where all the computation is performed ahead of time. Much of the work on universal algorithms has focused on TSP. For Euclidean TSP in the plane, Platzman and Bartholdi [49] gave an $O(\log n)$-approximate universal algorithm. Hajiaghayi et al. [31] generalized this result to an $O(\log^2 n)$-approximation for minor-free metrics, and Schalekamp and Shmoys [50] gave an $O(\log n)$-approximation for tree metrics. For arbitrary metrics, Jia et al. [35] presented an $O(\log^4 n/\log\log n)$-approximation, which improves to an $O(\log n)$-approximation for doubling metrics. The approximation factor for arbitrary metrics was improved to $O(\log^2 n)$ by Gupta et al. [26]. It is also known that these logarithmic bounds are essentially tight for universal TSP [9, 10, 22, 31]. For the metric Steiner tree problem, Jia et al. [35] adapted their own TSP algorithm to provide an $O(\log^4 n/\log\log n)$-approximate universal algorithm, which is also tight up to the exponent of the log [2, 10, 35]. Busch et al. [12] present an $O(2^{\sqrt{\log n}})$-approximation for universal Steiner tree on general graphs and an $O(\text{polylog}(n))$-approximation for minor-free graphs. Finally, for universal (weighted) set cover, Jia et al. [35] (see also Reference [23]) provide an $O(\sqrt{n\log n})$-approximate universal algorithm and an almost matching lower bound.

The problem of minimizing regret has been studied in the context of robust optimization. The robust 1-median problem was introduced for tree metrics by Kouvelis and Yu [42] and several faster algorithms and for general metrics were developed in the following years (e.g., see Reference [8]). For robust k-center, Averbakh and Berman [8] gave a reduction to a linear number of ordinary k-center problems, and thus for classes of instances where the ordinary k-center problem is polynomial time solvable (e.g., instances with constant k or on tree metrics) this problem is also polynomial time solvable [7]. A different notion of robust algorithms is one where a set S of possible scenarios is provided as part of the input to the problem. This model was originally considered for network design problems (see the survey by Chekuri [18]). Anthony et al. [3] gave an $O(\log n + \log |S|)$-approximation algorithm for solving k-median and a variety of related problems in this model (see also Reference [11]) on an n-point metric space. However, note that $|S|$ can be exponential in $|C|$ in general.

Another popular model for uncertainty is two-stage optimization (e.g., References [16, 19, 20, 27–29, 39, 51–53]). Here, the first stage presents a set of realizable instances (or a distribution over them) and the second stage chooses one of those realizations. The algorithm is free to make choices at either stage but those choices come at a higher cost in the second stage when it has more information about the input. Because of the different costs, results in this model have no bearing on our setting.

Roadmap. We present the constant approximation algorithms (Theorem 1.1) for universal k-median, ℓ_p-clustering (k-means is a special case), and k-center in Sections 2, 3, and 4, respectively. In describing these algorithms, we defer the clustering with discounts algorithms used in the rounding to Appendix C. We also give the extensions to universal clustering with fixed clients for k-median, k-means/ℓ_p-clustering, and k-center (Theorem 1.2) in Sections 5, 6, and 7. Finally, the hardness results for general metrics and for Euclidean metrics (Theorem 1.3) appear in Sections 8 and 9, respectively.

2 UNIVERSAL k-MEDIAN

In this section, we prove the following theorem:

Theorem 2.1. *There exists a* $(27, 49)$*-approximate universal algorithm for the* k*-median problem.*

We follow the recipe described in Section 1.2. Namely, the algorithm has two components. The first component is a separation oracle for the regret minimization LP based on submodular maximization, which we define below.

Submodular Maximization with Cardinality Constraints. A (non-negative) function $f : 2^E \to \mathbb{R}_0^+$ is said to be *submodular* if for all $S \subseteq T \subseteq E$ and $x \in E$, we have $f(T \cup \{x\}) - f(T) \leq f(S \cup \{x\}) - f(S)$. It is said to be *monotone* if for all $S \subseteq T \subseteq E$, we have $f(T) \geq f(S)$. The following theorem for maximizing monotone submodular functions subject to a cardinality constraint is well known.

Theorem 2.2 (Nemhauser et al. [48]). *For the problem of finding* $S \subseteq E$ *that maximizes a monotone submodular function* $f : 2^E \to \mathbb{R}_0^+$*, the natural greedy algorithm that starts with* $S = \emptyset$ *and repeatedly adds* $x \in E$ *that maximizes* $f(S \cup \{x\})$ *until* $|S| = k$*, is a* $\frac{e}{e-1} \approx 1.58$*-approximation.*

We give the reduction from the separation oracle to submodular maximization in Section 2.1 and then employ the above theorem. The second component of our framework is a rounding algorithm that employs the k-median with discounts problem, which we define below.

k-median with Discounts. In the k-median with discounts problem, we are given a k-median instance but where each client j has an additional (non-negative) parameter r_j called its *discount*. Just as in the k-median problem, our goal is to place k cluster centers that minimize the total connection costs of all clients. But the connection cost for client j can now be discounted by up to r_j, i.e., client j with connection cost c_j contributes $(c_j - r_j)^+ := \max\{0, c_j - r_j\}$ to the objective of the solution.

Let OPT be the cost of an optimal solution to the k-median with discounts problem. We say an algorithm ALG that outputs a solution with connection cost c_j for client j is a (γ, σ)-approximation if

$$\sum_{j \in C} (c_j - \gamma \cdot r_j)^+ \leq \sigma \cdot \text{OPT}.$$

That is, a (γ, σ)-approximate algorithm outputs a solution whose objective function when computed using discounts $\gamma \cdot r_j$ for all j is at most σ times the optimal objective using discounts r_j. In

the case where all r_j are equal, Reference [25] gave a $(9, 6)$-approximation algorithm for this problem based on the classic primal-dual algorithm for k-median. The following lemma generalizes their result to the setting where the r_j may differ:

Lemma 2.3. *There exists a (deterministic) polynomial-time* $(9, 6)$*-approximation algorithm for the* k*-median with discounts problem.*

We give details of the algorithm and the proof of this lemma in Appendix C. We note that when all r_j are equal, the constants in Reference [25] can be improved (see, e.g., Reference [15]); we do not know of any similar improvement when the r_j may differ. In Section 2.2, we give the reduction from rounding the fractional solution for universal k-median to the k-median with discounts problem and then employ the above lemma.

2.1 Universal k-median: Fractional Algorithm

The standard k-median polytope (see, e.g., Reference [34]) is given by

$$P = \left\{ (x, y) : \sum_i x_i \leq k; \forall i, j : y_{ij} \leq x_i; \forall j : \sum_i y_{ij} \geq 1; \forall i, j : x_i, y_{ij} \in [0, 1] \right\}.$$

Here x_i represents whether point i is chosen as a cluster center, and y_{ij} represents whether client j connects to i as its cluster center. Now, consider the following LP formulation for minimizing regret r:

$$\min \left\{ r : (x, y) \in P; \forall C' \subseteq C : \sum_{j \in C'} \sum_i c_{ij} y_{ij} - \text{OPT}(C') \leq r \right\}, \tag{1}$$

where $\text{OPT}(C')$ is the cost of the (integral) optimal solution in realization C'. Note that the new constraints, $\forall C' \subseteq C : \sum_{j \in C'} \sum_i c_{ij} y_{ij} - \text{OPT}(C') \leq r$ (we call it the regret constraint set), require that the regret is at most r in all realizations.

To solve LP (1), we need a separation oracle for the regret constraint set. Note that there are exponentially many constraints corresponding to realizations C'; moreover, even for a single realization C', computing $\text{OPT}(C')$ is **NP**-hard. So we resort to designing an *approximate* separation oracle. Fix some fractional solution (x, y, r). Overloading notation, let $S(C')$ denote the cost of the solution with cluster centers S in realization C'. By definition, $\text{OPT}(C') = \min_{S \subseteq F, |S| = k} S(C')$. Then designing a separation oracle for the regret constraint set is equivalent to determining if the following inequality holds:

$$\max_{C' \subseteq C} \max_{S \subseteq F, |S| = k} \left[\sum_{j \in C'} \sum_i c_{ij} y_{ij} - S(C') \right] \leq r.$$

We flip the order of the two maximizations and define $f_y(S)$ as follows:

$$f_y(S) = \max_{C' \subseteq C} \left[\sum_{j \in C'} \sum_i c_{ij} y_{ij} - S(C') \right].$$

Then designing a separation oracle is equivalent to maximizing $f_y(S)$ for $S \subseteq F$ subject to $|S| = k$. The rest of the proof consists of showing that this function is monotone, submodular, and efficiently computable.

Lemma 2.4. *Fix* y*. Then,* $f_y(S)$ *is a monotone submodular function in* S*. Moreover,* $f_y(S)$ *is efficiently computable for a fixed* S*.*

Proof. Let $d(j,S) := \min_{i' \in S} c_{i'j}$ denote the distance from client j to the nearest cluster center in S. If $S = \emptyset$, then we say $d(j,S) := \infty$. The value of C' that defines $f_y(S)$ is the set of all clients closer to S than to the fractional solution y, i.e., $\sum_i c_{ij} y_{ij} > \min_{i' \in S} c_{i'j}$. This immediately establishes efficient computability of $f_y(S)$. Moreover, we can equivalently write $f_y(S)$ as follows:

$$f_y(S) = \sum_{j \in C} \left(\sum_i c_{ij} y_{ij} - d(j,S) \right)^+ .$$

A sum of monotone submodular functions is a monotone submodular function, so it suffices to show that for all clients j, the new function $g_{y,j}(S) := (\sum_i c_{ij} y_{ij} - d(j,S))^+$ is monotone submodular.

- $g_{y,j}$ is *monotone*: For $S \subseteq T$, $d(j,T) \leq d(j,S)$, and thus $(\sum_i c_{ij} y_{ij} - d(j,S))^+ \leq (\sum_i c_{ij} y_{ij} - d(j,T))^+$.
- $g_{y,j}$ is submodular if

$$\forall S \subseteq T \subseteq F, \forall x \in F : g_{y,j}(S \cup \{x\}) - g_{y,j}(S) \geq g_{y,j}(T \cup \{x\}) - g_{y,j}(T).$$

  Fix S, T, and x. Assume $g_{y,j}(T \cup \{x\}) - g_{y,j}(T)$ is positive (if it is zero, by monotonicity the above inequality trivially holds). This implies that x is closer to client j than any cluster center in T (and hence S, too), i.e., $d(j,x) \leq d(j,T) \leq d(j,S)$. Thus, $d(j,x) = d(j, S \cup \{x\}) = d(j, T \cup \{x\})$, which implies that $g_{y,j}(S \cup \{x\}) = g_{y,j}(T \cup \{x\})$. Then we just need to show that $g_{y,j}(S) \leq g_{y,j}(T)$, but this holds by monotonicity. □

By standard results (see, e.g., GLS [24]), we get an (α, β)-approximate fractional solution for universal k-median via the ellipsoid method if we have an approximate separation oracle for LP (1) that given a fractional solution (x,y,r) does either of the following:

- declares (x,y,r) feasible, in which case (x,y) has cost at most $\alpha \cdot \text{OPT}(\mathbf{I}) + \beta \cdot r$ in all realizations or
- putputs an inequality violated by (x,y,r) in LP (1).

The approximate separation oracle does the following for the regret constraint set (all other constraints can be checked exactly): Given a solution (x,y,r), find an $\frac{e-1}{e}$-approximate maximizer S of f_y via Lemma 2.4 and Theorem 2.2. Let C' be the set of clients closer to S than the fractional solution y (i.e., the realization that maximizes $f_y(S)$). If $f_y(S) > r$, then the separation oracle returns the violated inequality $\sum_{j \in C'} \sum_i c_{ij} y_{ij} - S(C') \leq r$; else, it declares the solution feasible. Whenever the actual regret of (x,y) is at least $\frac{e}{e-1} \cdot r$, this oracle will find S such that $f_y(S) > r$ and output a violated inequality. Hence, we get the following lemma:

Lemma 2.5. *There exists a deterministic algorithm that in polynomial time computes a fractional* $\frac{e}{e-1} \approx 1.58$*-approximate solution for* LP (1) *representing the universal* k*-median problem.*

2.2 Universal k-Median: Rounding Algorithm

Let FRAC denote the $\frac{e}{e-1}$-approximate fractional solution to the universal k-median problem provided by Lemma 2.5. We will use the following property of k-median, shown by Archer et al. [4].

Lemma 2.6 ([4]). *The integrality gap of the natural* LP *relaxation of the* k*-median problem is at most* 3.

Lemmas 2.5 and 2.6 imply that that for any set of clients C',

$$\frac{1}{3} \cdot \text{OPT}(C') \leq \text{FRAC}(C') \leq \text{OPT}(C') + \frac{e}{e-1} \cdot \text{MR}. \tag{2}$$

Our overall goal is to obtain a solution SOL that minimizes $\max_{C' \subseteq C} [\text{SOL}(C') - \text{OPT}(C')]$. But instead of optimizing over the exponentially many different $\text{OPT}(C')$ solutions, we use the surrogate $3 \cdot \text{FRAC}(C')$, which has the advantage of being defined by a fixed solution FRAC but still 3-approximates $\text{OPT}(C')$ by Equation (2). This suggests minimizing the following objective instead: $\max_{C'} [\text{SOL}(C') - 3 \cdot \text{FRAC}(C')]$. For a given solution SOL, the set of clients C' that maximizes the new expression are the clients whose connection costs in SOL (denoted c_j) exceeds 3 times their cost in FRAC (denoted f_j):

$$\max_{C'} [\text{SOL}(C') - 3 \cdot \text{FRAC}(C')] = \sum_{j \in C} (c_j - 3f_j)^+.$$

But minimizing this objective is precisely the aim of the k-median with discounts problem, where the discount for client j is $3f_j$. This allows us to invoke Lemma 2.3 for the k-median with discounts problem.

Thus, our overall algorithm is as follows. First, use Lemma 2.5 to find a fractional solution $\text{FRAC} = (x, y, r)$. Let $f_j := \sum_i c_{ij} y_{ij}$ be the connection cost of client j in FRAC. Then, construct a k-median with discounts instance where client j has discount $3f_j$, and use Lemma 2.3 on this instance to obtain the final solution to the universal k-median problem.

We now complete the proof of Theorem 2.1 using the above lemmas.

Proof of Theorem 2.1. Let m_j be the connection cost of MRS to client j. Then,

$$\begin{aligned} \text{MR} = \max_{C' \subseteq C} [\text{MRS}(C') - \text{OPT}(C')] &\geq \max_{C' \subseteq C} [\text{MRS}(C') - 3 \cdot \text{FRAC}(C')] \quad \text{(by Equation (2))} \\ &= \sum_{j \in C : m_j > 3f_j} (m_j - 3f_j) = \sum_{j \in C} (m_j - 3f_j)^+. \end{aligned}$$

Thus, MR upper bounds the optimal objective in the k-median with discounts instance that we construct. Let c_j be the connection cost of client j in the solution output by the algorithm. Then, by Lemma 2.3, we get that

$$\sum_{j \in C} (c_j - 27f_j)^+ \leq 6 \cdot \sum_{j \in C} (m_j - 3f_j)^+ \leq 6 \cdot \text{MR}. \tag{3}$$

As a consequence, we have

$$\forall C' \subseteq C : \sum_{j \in C'} c_j = \sum_{j \in C'} [27f_j + (c_j - 27f_j)] \leq \sum_{j \in C'} 27f_j + \sum_{j \in C'} (c_j - 27f_j)^+ \leq 27 \cdot \text{FRAC}(C') + 6 \cdot \text{MR},$$

where the last step uses the definition of f_j and Equation (3). Now, using the bound on $\text{FRAC}(C')$ from Equation (2) in the inequality above, we have the desired bound on the cost of the algorithm,

$$\forall C' \subseteq C : \sum_{j \in C'} c_j \leq 27 \cdot \text{FRAC}(C') + 6 \cdot \text{MR} \leq 27 \left[\text{OPT}(C') + \frac{e}{e-1} \cdot \text{MR} \right] + 6 \cdot \text{MR} \leq 27 \cdot \text{OPT}(C') + 49 \cdot \text{MR}.$$

□

3 UNIVERSAL ℓ_p-CLUSTERING AND UNIVERSAL k-MEANS

In this section, we give universal algorithms for ℓ_p-clustering with the following guarantee:

Theorem 3.1. *For all $p \geq 1$, there exists a $(54p, 103p^2)$-approximate universal algorithm for the ℓ_p-clustering problem.*

As a corollary, we obtain the following result for universal k-means ($p = 2$).

Corollary 3.2. *There exists a $(108, 412)$-approximate universal algorithm for the k-means problem.*

Before describing further details of the universal ℓ_p-clustering algorithm, we note a rather unusual feature of the universal clustering framework. Typically, in standard ℓ_p-clustering, the algorithms effectively optimize the ℓ_p^p objective (e.g., sum of squared distances for k-means) because these are equivalent in the following sense: An α-approximation for the ℓ_p objective is equivalent to an α^p-approximation for the ℓ_p^p objective. But this equivalence fails in the setting of universal algorithms for reasons that we discuss below. Indeed, we first give a universal ℓ_p^p-clustering algorithm, which is a simple extension of the k-median algorithm, and then give universal ℓ_p-clustering algorithms, which turns out to be much more challenging.

3.1 Universal ℓ_p^p-Clustering

As in universal k-median, we can write an LP formulation for universal ℓ_p^p-clustering, i.e., clustering with the objective $\text{SOL}(C') = \sum_{j \in C'} \text{COST}(j, \text{SOL})^p$:

$$\min \left\{ r : (x, y) \in P; \forall C' \subseteq C : \sum_{j \in C'} \sum_i c_{ij}^p y_{ij} - \text{OPT}(C') \leq r \right\}, \tag{4}$$

where P is still the k-median polytope defined in Section 2.1.

The main difficulty is that the ℓ_p^p distances no longer form a metric, i.e., do not satisfy triangle inequality. Nevertheless, the distances still have a metric connection that they are the pth power of metric distances. We show that this connection is sufficient to prove the following result:

THEOREM 3.3. *For all $p \geq 1$, there exists a $(27^p, 27^p \cdot \frac{e}{e-1} + \frac{2}{3} \cdot 9^p)$-approximate algorithm for the universal ℓ_p^p clustering problem.*

As in universal k-median, a key component in proving Theorem 3.3 is a rounding algorithm that employs a bi-criteria approximation to the *ℓ_p^p-clustering with discounts* problem. Indeed, this result will also be useful in the next subsection, when we consider the universal ℓ_p-clustering problem. So, we formally define ℓ_p^p-clustering with discounts problem below and state our result for it.

ℓ_p^p-clustering with Discounts. In this problem, are given a ℓ_p^p-clustering instance but where each client j has an additional (non-negative) parameter r_j called its *discount*. Our goal is to place k cluster centers that minimize the total connection costs of all clients. But the connection cost for client j can now be discounted by up to r_j^p, i.e., client j with connection cost c_j contributes $(c_j^p - r_j^p)^+ := \max\{0, c_j^p - r_j^p\}$ to the objective of the solution. (Note that the k-median with discounts problem that we described in the previous section is a special case of this problem for $p = 1$.)

Let OPT be the cost of an optimal solution to the ℓ_p^p-clustering with discounts problem. We say an algorithm ALG that outputs a solution with connection cost c_j for client j is a (γ^p, σ)-approximation[5] if

$$\sum_{j \in C} (c_j^p - \gamma^p \cdot r_j^p)^+ \leq \sigma \cdot \text{OPT}.$$

That is, a (γ^p, σ)-approximate algorithm outputs a solution whose objective function computed using discounts $\gamma \cdot r_j$ for all j is at most σ times the optimal objective using discounts r_j. We give the following result about the ℓ_p^p-clustering with discounts problem (see Appendix C for details):

LEMMA 3.4. *There exists a (deterministic) polynomial-time $(9^p, \frac{2}{3} \cdot 9^p)$-approximation algorithm for the ℓ_p^p-clustering with discounts problem.*

[5]We refer to this as a (γ^p, σ)-approximation instead of a (γ, σ)-approximation to emphasize the difference between the scaling factor for discounts γ and the loss in approximation factor γ^p.

We now employ this lemma in obtaining Theorem 3.3. Recall that the universal k-median result in the previous section had three main ingredients:

- Lemma 2.5 to obtain an $\frac{e}{e-1}$-approximate fractional solution. This continues to hold for the ℓ_p^p objective, since Lemma 2.5 does not use any metric property.
- An upper bound of 3 on the integrality gap of the natural LP relaxation of k-median from Reference [4]. The same result now gives an upper bound of 3^p on the integrality gap of ℓ_p^p-clustering.
- Lemma 2.3 to obtain an approximation guarantee for the k-median with discounts problem. This is where the metric property of the connection costs in the k-median problem was being used. Nevertheless, Lemma 3.4 above gives a generalization of Lemma 2.3 to the ℓ_p^p-clustering with discounts problem.

Theorem 3.3 now follows from these three observations using exactly the same steps as Theorem 2.1 in the previous section; we omit these steps for brevity. □

The rest of this section is dedicated to the universal ℓ_p-clustering problem. As for k-median, we have two stages, the fractional algorithm and the rounding algorithm, which we present in the next two subsections.

3.2 Universal ℓ_p-Clustering: Fractional Algorithm

Let us start by describing the fractional relaxation of the universal ℓ_p-clustering problem[6] (again, P is the k-median polytope defined as in Section 2.1),

$$\min \left\{ r : (x,y) \in P; \forall C' \subseteq C : \left(\sum_{j \in C'} \sum_i c_{ij}^p y_{ij} \right)^{1/p} - \textsc{opt}(C') \leq r \right\}, \tag{5}$$

As described earlier, when minimizing regret, the ℓ_p and ℓ_p^p objectives are no longer equivalent. For instance, recall that one of the key steps in Lemma 2.5 was to establish the submodularity of the function $f_y(S)$ denoting the maximum regret caused by any realization when comparing two given solutions: a fractional solution y and an integer solution S. Indeed, the worst-case realization had a simple structure: Choose all clients that have a smaller connection cost for S than for y. This observation continues to hold for the ℓ_p^p objective because of the linearity of $f_y(S)$ as a function of the realized clients once y and S are fixed. But the ℓ_p objective is not linear even after fixing the solutions, and as a consequence, we lose both the simple structure of the maximizing realization as well as the submodularity of the overall function $f_y(S)$. For instance, consider two clients: one at distances 1 and 0 and another at distances $1+\epsilon$ and 1 from y and S, respectively. Using the ℓ_p objective, the regret with both clients is $(2+\epsilon)^{1/p} - 1$, whereas with just the first client the regret is 1, which is larger for $p \geq 2$.

The above observation results in two related difficulties: first, that $f_y(S)$ is not submodular and hence standard submodular maximization techniques do not apply, but also that given y and S, we cannot even compute the function $f_y(S)$ efficiently. To overcome this difficulty, we further refine the function $f_y(S)$ to a collection of functions $f_{y,Y}(S)$ by also fixing the cost of the fractional solution y to at most a given value Y. As we will soon see, this allows us to relate the ℓ_p objective to the ℓ_p^p objective but under an additional "knapsack"-like packing constraint. It is still not immediate

[6]The constraints are not simultaneously linear in y and r, although fixing r, we can write these constraints as $\sum_{j \in C'} \sum_i c_{ij}^p y_{ij} \leq (\textsc{opt}(C') + r)^p$, which is linear in y. In turn, to solve this program we bisection search over r, using the ellipsoid method to determine if there is a feasible point for each fixed r.

that $f_{y,Y}$ is efficiently computable because of the knapsack constraint that we have introduced. Our second observation is that relaxing the (**NP**-hard) integer knapsack problem to the corresponding (poly-time) *fractional* knapsack problem does not affect the optimal value of $f_{y,Y}(S)$ (i.e., allowing fractional clients does not increase y's regret), while making the function efficiently computable. As a bonus, the relaxation to fractional knapsack also restores submodularity of the function, allowing us to use standard maximization tools as earlier. We describe these steps in detail below.

To relate the regret in the ℓ_p and ℓ_p^p objectives, let $\textsc{frac}_p^q(C')$ and $S_p^q(C')$ denote the ℓ_p objective to the qth power for y and S, respectively, in realization C' (and let $\textsc{frac}_p(C'), S_p(C')$ denote the corresponding ℓ_p objectives). Assume that y's regret against S is non-zero. Then

$$\begin{aligned}
\max_{C'\subseteq C}\left[\textsc{frac}_p(C') - S_p(C')\right] &= \max_{C'\subseteq C}\left[\frac{\textsc{frac}_p^p(C') - S_p^p(C')}{\sum_{q=0}^{p-1}\textsc{frac}_p^q(C')S_p^{p-1-q}(C')}\right]\\
&\simeq_p \max_{C'\subseteq C}\left[\frac{\textsc{frac}_p^p(C') - S_p^p(C')}{\textsc{frac}_p^{p-1}(C')}\right]\\
&= \max_{Y}\max_{C'\subseteq C:\textsc{frac}_p^p(C')\le Y}\left[\frac{\textsc{frac}_p^p(C') - S_p^p(C')}{Y^{1-1/p}}\right]\\
&= \max_{Y}\left[\frac{\max_{C'\subseteq C,\,\textsc{frac}_p^p(C')\le Y}\left[\textsc{frac}_p^p(C') - S_p^p(C')\right]}{Y^{1-1/p}}\right].
\end{aligned}$$

The $\simeq_p$ denotes equality to within a factor of p and uses the fact that if the regret is non-zero; then for every C' such that $\textsc{frac}_p(C') > S_p(C')$ (one of which is always the maximizer of all expressions in this equation), every term in the sum in the denominator is upper bounded by $\textsc{frac}_p^{p-1}(C')$.

We would like to argue that the numerator,

$$\max\{\textsc{frac}_p^p(C') - S_p^p(C') : C'\subseteq C, \textsc{frac}_p^p(C')\le Y\},$$

is a submodular function of S. If we did this, then we could find an adversary and realization of clients that (approximately) maximizes the regret of y by iterating over all (discretized) values of Y. But, as described above, it is easier to work with its fractional analog because of the knapsack constraint,

$$f_{y,Y}(S) := \max\{\textsc{frac}_p^p(\mathbf{I}) - S_p^p(\mathbf{I}) : \mathbf{I}\in[0,1]^C, \textsc{frac}_p^p(\mathbf{I})\le Y\}.$$

Here, $\textsc{frac}_p^p(\mathbf{I}) := \sum_{j\in C} d_j\cdot\sum_{i\in F} c_{ij}^p y_{ij}$ and $S_p^p(\mathbf{I}) := \sum_{j\in C} d_j\cdot\min_{i\in S} c_{ij}^p$ are the natural extensions of the ℓ_p^p objective to fractional clients, where d_j is the fraction of client j that is included in $\mathbf{I}$. The next lemma shows that allowing fractional clients does not affect the maximum regret:

Lemma 3.5. *For any two solutions y, S, there exists a global maximum of $\textit{FRAC}_p(\mathbf{I}) - S_p(\mathbf{I})$ over $\mathbf{I}\in[0,1]^C$ where all the clients are integral, i.e., $\mathbf{I}\in\{0,1\}^C$. Therefore,*

$$\max_{\mathbf{I}\in[0,1]^C}\left[\textit{FRAC}_p(\mathbf{I}) - S_p(\mathbf{I})\right] = \max_{C'\subseteq C}\left[\textit{FRAC}_p(C') - S_p(C')\right].$$

We remark that, unlike for the ℓ_p^p objective, integrality of the maximizer is not immediate for the ℓ_p objective because the regret of y compared to S is not a linear function of $\mathbf{I}$.

Proof. We will show that the derivative of $\textsc{frac}_p(\mathbf{I}) - S_p(\mathbf{I})$ when $\textsc{frac}_p(\mathbf{I}) > S_p(\mathbf{I})$ with respect to a fixed d_j is either always positive or always negative for $d_j\in(0,1)$ or negative while $0 < d_j < d'$ and then positive afterwards. This gives that any $\mathbf{I}$ with a fractional coordinate where

$\text{FRAC}_p(\mathbf{I}) > S_p(\mathbf{I})$ (which is necessary for a fractional $\mathbf{I}$ to be a global maximum but not the integral all-zeroes vector) cannot be a local maximum, giving the lemma.

To show this property of the derivative, letting $\mathbf{I}_{-j}$ denote $\mathbf{I}$ with $d_j = 0$, we have $\text{FRAC}_p(\mathbf{I}) - S_p(\mathbf{I}) = (\text{FRAC}_p^p(\mathbf{I}_{-j}) + c_1 d_j)^{1/p} - (S_p^p(\mathbf{I}_{-j}) + c_2 d_j)^{1/p}$, where c_1, c_2 are the ℓ_p^p distance from j to FRAC, S respectively. For positive d_j, the derivative with respect to d_j is well defined and equals

$$\frac{1}{p}\left(\frac{c_1}{\left(\text{FRAC}_p^p(\mathbf{I}_{-j}) + c_1 d_j\right)^{1-1/p}} - \frac{c_2}{\left(S_p^p(\mathbf{I}_{-j}) + c_2 d_j\right)^{1-1/p}}\right).$$

The derivative is positive if the following inequality holds:

$$\frac{S_p^p(\mathbf{I}_{-j}) + c_2 d_j}{\text{FRAC}_p^p(\mathbf{I}_{-j}) + c_1 d_j} > \left(\frac{c_2}{c_1}\right)^{\frac{p}{p-1}}.$$

We first focus on the case where $c_1 > c_2$. The left-hand side starts at $S_p^p(\mathbf{I}_{-j})/\text{FRAC}_p^p(\mathbf{I}_{-j})$ and monotonically and asymptotically approaches c_2/c_1 as d_j increases. This implies that either it is always at least c_2/c_1 or it is increasing and approaching c_2/c_1, i.e., at least $(c_2/c_1)^{p/(p-1)}$ for $d_j > 0$ or less than $(c_2/c_1)^{p/(p-1)}$ for all $d_j < d'$ for some d' and then greater than $(c_2/c_1)^{p/(p-1)}$ for all $d_j > d'$ (here, we are using the fact that $c_1 > c_2$ and so $(c_2/c_1)^{p/(p-1)} < c_2/c_1$). In turn, the desired property of the derivative holds.

In the case where $c_1 \leq c_2$, at any optimum $S_p^p(\mathbf{I}_{-j}) + c_2 d_j < \text{FRAC}_p^p(\mathbf{I}_{-j}) + c_1 d_j$ (otherwise $\text{FRAC}_p^p(\mathbf{I}) < S_p^p(\mathbf{I})$ and so $\mathbf{I}$ cannot be a maximum because the all zeroes vector achieves a better objective) and so the derivative is always negative in this case as desired. □

It turns out that relaxing to fractional clients not only helps in efficient computability of the function $f_{y,Y}(S)$ but also simplifies the proof of submodularity of the function.

Lemma 3.6. *The function $f_{y,Y}(S)$ as defined above is submodular.*

Proof. Fix a universal clustering instance, fractional solution y, and value Y. Consider any $S \subseteq T \subseteq F$ and $x \in F$. $f_{y,Y}(T \cup \{x\}) - f_{y,Y}(T) \leq f_{y,Y}(S \cup \{x\}) - f_{y,Y}(S)$. $f_{y,Y}(S)$ is the optimum of a fractional knapsack instance where each client is an item with value equal to the difference between its contribution to $\text{FRAC}_p^p(\mathbf{I}) - S_p^p(\mathbf{I})$ and weight equal to its contribution to $\text{FRAC}_p^p(\mathbf{I})$, with the knapsack having total weight Y. For simplicity, we can assume there is a dummy item with value 0 and weight Y in the knapsack instance as well (a value 0 item cannot affect the optimum). Note that the weights are fixed for any S, and the values increase monotonically with S. We will refer to the latter fact as *monotonicity of values*. The optimum of a fractional knapsack instance is given by sorting the items in decreasing ratio of value to weight and taking the (fractional) prefix of the items sorted this way that has total weight Y. We will refer to this fact as the *prefix property*. We will show that we can construct a fractional knapsack solution that when using cluster centers $S \cup \{x\}$ has value at least $f_{y,Y}(S) + f_{y,Y}(T \cup \{x\}) - f_{y,Y}(T)$, proving the lemma. For brevity, we will refer to fractions of clients that may be in the knapsack as if they were integral clients.

Consider $f_{y,Y}(T \cup \{x\}) - f_{y,Y}(T)$. We can split this difference into four cases as follows:

(1) A client in the optimal knapsack for S, T, and $T \cup \{x\}$.
(2) A client in the optimal knapsack for S and $T \cup \{x\}$ but not for T.
(3) A client in the optimal knapsack for T and $T \cup \{x\}$ but not for S.
(4) A client in the optimal knapsack for $T \cup \{x\}$ but not for S or T.

In every case, the client's value must have increased (otherwise, it cannot contribute to the difference in cases 1 and 3, or it must also be in T's knapsack in cases 2 and 4), i.e., x is the closest cluster

center to the client in $T \cup \{x\}$ (and thus $S \cup \{x\}$). Let w_1, w_2, w_3, w_4 be the total weight of clients in each case. The total weight of clients in T's knapsack but not $T \cup \{x\}$ is $w_2 + w_4$. We will refer to these clients as replaced clients. The increase in value due to cases 2 and 4 can be thought of as replacing the replaced clients with case 2 and 4 clients. In particular, we will think of the case 2 clients as replacing the replaced clients of weight w_2 with the smallest total value for T, and the case 4 clients as replacing the remaining replaced clients (i.e., those with the largest total value for T).

Without loss of generality, we assume there are no case 1 clients. By the prefix property, any of the knapsack instances for $S, T, T \cup \{x\}$ (and also $S \cup \{x\}$ by monotonicity of values and the prefix property) has optimal value equal to the total value of case 1 clients plus the optimal value of a smaller knapsack instance with total weight $Y - w_1$ and all clients except case 1 clients available. The value of case 1 clients for $S \cup \{x\}$ and $T \cup \{x\}$ is the same (since the values are determined by x), and can only be smaller for S than T by monotonicity of values. In turn, we just need to construct a knapsack for $S \cup \{x\}$ for the smaller instance with no case 1 clients whose value is at least that of S plus the contribution to $f_{y,Y}(T \cup \{x\}) - f_{y,Y}(T)$ from cases 2–4.

To build the desired knapsack for $S \cup \{x\}$, we start with the knapsack for S. The case 2 clients in S's knapsack by the prefix property have less value for S than for T by monotonicity of values. By the prefix property, for T the case 2 clients have less value than the replaced clients of total weight w_2 with the smallest total value for T (since the former are not in the optimal knapsack for T, and the latter are). So, the increase in value of the case 2 clients in S's knapsack is at least the contribution to $f_{y,Y}(T \cup \{x\}) - f_{y,Y}(T)$ due to replacing clients in T's knapsack with case 2 clients.

To account for the case 4 clients, we take the clients in S's knapsack that are not case 2 clients with weight w_4 and the least total value for T and replace them with the case 4 clients. These clients are among the clients of total weight $w_2 + w_4$ with the lowest value-to-weight ratios (for T) in S's knapsack (they are not necessarily the clients of total weight w_4 with the lowest value-to-weight ratios, because we chose not to include case 2 clients in this set). However, the replaced clients in T's knapsack all have value-to-weight ratios for T greater than at least w_2 weight of other clients in T's knapsack (those replaced by the case 2 clients). So by monotonicity of values and the prefix property, the clients we replace in S's knapsack with case 4 clients have lower value for S than the clients being replaced by case 4 clients do for T, and so we increase the value of our knapsack by more than the contribution to $f_{y,Y}(T \cup \{x\}) - f_{y,Y}(T)$ due to case 4 clients.

Last, we take any clients in S's knapsack that are not in T's knapsack or case 2 or case 4 clients with total weight w_3 and replace them with the case 3 clients. Since these clients are not in T's knapsack, their value for T (and thus their value for S by monotonicity of values) is less than the case 3 clients' value for T by the prefix property. In turn, this replacement increases the value of our knapsack by more than the contribution to $f_{y,Y}(T \cup \{x\}) - f_{y,Y}(T)$ due to case 3 clients. □

This lemma allows us to now give an approximate separation oracle for fractional solutions of universal ℓ_p-clustering by trying all guesses for Y (ℓ_p-SepOracle in Figure 1).

One complication of using the above as a separation oracle in the ellipsoid algorithm is that it outputs linear constraints, whereas the actual constraints in the fractional relaxation given in Equation (5) are non-linear. So, in the following lemma, we need some additional work to show that violation of the linear constraints output by ℓ_p-SepOracle also implies violation of the non-linear constraints in Equation (5).

Lemma 3.7. *For any $p \geq 1, \epsilon > 0$ there exists an algorithm that finds a $(\frac{e}{e-1} \cdot p + \epsilon)$-approximation to the LP for the universal ℓ_p-clustering problem.*

Proof. The algorithm is to use the ellipsoid method with ℓ_p-SepOracle as the separation oracle. We note that $f_{y,Y}(S)$ can be evaluated in polynomial time by computing optimal fractional knapsacks as discussed in the proof of Lemma 3.6. In addition, there are polynomially values of Y

ℓ_p-SepOracle$((x, y, r), F, C)$:
Input: Fractional solution (x, y, r), set of cluster centers F, set of all clients C

1: **if** Any constraint in (5) except the regret constraint set is violated **then**
2: **return** the violated constraint
3: **end if**
4: $c_{\min} \leftarrow \min_{i\in F, j\in C} c_{ij}^p$, $c_{\max} \leftarrow \sum_{j\in C} \max_{i\in F} c_{ij}^p$
5: **for** $Y \in \{c_{\min}, c_{\min}(1+\epsilon'), c_{\min}(1+\epsilon')^2, \ldots c_{\min}(1+\epsilon')^{\lceil \log_{1+\epsilon'} c_{\max}/c_{\min}\rceil}\}$ **do**
6: $S \leftarrow \frac{e-1}{e}$-maximizer of $f_{y,Y}$ subject to $|S| \le k$ via Theorem 2.2
7: $\mathbf{I}' \leftarrow \text{argmax}_{\mathbf{I}\in[0,1]^C : \sum_{j\in C} d_j \sum_{i\in F} c_{ij}^p y_{ij} \le Y} \sum_{j\in C} d_j \sum_{i\in F} c_{ij}^p y_{ij} - \sum_{j\in C} d_j \min_{i\in S} c_{ij}^p$
8: **if** $\frac{1}{pY^{1-1/p}} \left[\sum_{j\in C} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - \sum_{j\in C} d'_j \min_{i\in S} c_{ij}^p\right] > r$ **then**
9: **return** $\frac{1}{pY^{1-1/p}} \left[\sum_{j\in C} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - \sum_{j\in C} d'_j \min_{i\in S} c_{ij}^p\right] \le r$
10: **end if**
11: **end for**
12: **return** "Feasible"

Fig. 1. Separation Oracle for ℓ_p-clustering.

that are iterated over, since $\lceil \log_{1+\epsilon'} c_{\max}/c_{\min}\rceil$ is $O(p \log n/\epsilon')$ times the number of bits needed to describe the largest value of c_{ij}. So each call to ℓ_p-SepOracle takes polynomial time.

By Lemma 3.5 and the observation that $\text{FRAC}_p^p(\mathbf{I}) - S_p^p(\mathbf{I}) \le p \cdot \text{FRAC}_p^{p-1}(\mathbf{I})(\text{FRAC}_p(\mathbf{I}) - S_p(\mathbf{I}))$ if $\text{FRAC}_p(\mathbf{I}) - S_p(\mathbf{I}) > 0$, then we get that ℓ_p-SepOracle cannot output a violated inequality if the input solution is feasible to Equation (5). So we just need to show that if $\text{FRAC}_p(C') - S_p(C') \ge (\frac{e}{e-1}p + \epsilon)r$ for some S, C', then ℓ_p-SepOracle outputs a violated inequality, i.e., does not output "Feasible." Let Y' be the smallest value of Y iterated over by ℓ_p-SepOracle that is at least $\text{FRAC}_p^p(C')$, and S' the $\frac{e-1}{e}$-maximizer of $f_{y,Y'}$ found by ℓ_p-SepOracle. We have for the $\mathbf{I}'$ found on Line 7 of ℓ_p-SepOracle:

$$\frac{1}{p(Y')^{1-1/p}} \left[\text{FRAC}_p^p(\mathbf{I}') - S'^p_p(\mathbf{I}')\right] \ge \frac{e-1}{pe(Y')^{1-1/p}} \max_{S, \mathbf{I}:\text{FRAC}_p^p(\mathbf{I}) \le Y'} \left[\text{FRAC}_p^p(\mathbf{I}') - S_p^p(\mathbf{I}')\right]$$

$$\overset{(i)}{\ge} \frac{e-1}{pe(Y')^{1-1/p}} \max_{S, \mathbf{I}:\text{FRAC}_p^p(\mathbf{I}) \le \text{FRAC}_p^p(C')} \left[\text{FRAC}_p^p(\mathbf{I}) - S_p^p(\mathbf{I})\right] \ge \frac{e-1}{pe(Y')^{1-1/p}} \left[\text{FRAC}_p^p(C') - S_p^p(C')\right]$$

$$\ge \frac{e-1}{pe(1+\epsilon')} \frac{\text{FRAC}_p^p(C') - S_p^p(C')}{\text{FRAC}_p^{p-1}(C')} = \frac{e-1}{pe(1+\epsilon')} \left[\text{FRAC}_p(C') - S_p(C')\right] > r.$$

In (i), we use the fact that for a fixed S, $\max_{\mathbf{I}:\text{FRAC}_p^p(\mathbf{I}) \le Y} [\text{FRAC}_p^p(\mathbf{I}') - S_p^p(\mathbf{I}')]$ is the solution to a fractional knapsack problem with weight Y and that decreasing the weight allowed in a fractional knapsack instance can only reduce the optimum. For the last inequality to hold, we just need to choose $\epsilon' < \epsilon \frac{(e-1)}{pe}$ in ℓ_p-SepOracle for the desired ϵ. This shows that for Y', ℓ_p-SepOracle will output a violated inequality as desired. □

3.3 Universal ℓ_p-Clustering: Rounding Algorithm

At a high level, we use the same strategy for rounding the fractional ℓ_p-clustering solution as we did with k-median. Namely, we solve a discounted version of the problem where the discount for each client is equal to the (scaled) cost of the client in the fractional solution. However, if we apply this directly to the ℓ_p objective, then we run into several problems. In particular, the linear discounts

are incompatible with the non-linear objective defined over the clients. A more promising idea is to use these discounts on the ℓ_p^p objective, which in fact is defined as a linear combination over the individual client's objectives. But, for this to work, we will first need to relate the regret bound in the ℓ_p^p objective to that in the ℓ_p objective. This is clearly not true in general, i.e., for all realizations. However, we show that the realization that maximizes the regret of an algorithm ALG against a fixed solution SOL in both objectives is the same under the following "farness" condition: *For every client, either ALG's connection is smaller than SOL's or it is at least p times as large as SOL's.* Given any solution SOL, it is easy to define a *virtual* solution $\widetilde{\text{SOL}}$ whose individual connection costs are bounded by p times that in SOL, and $\widetilde{\text{SOL}}$ satisfies the farness condition. This allows us to relate the regret of ALG against $\widetilde{\text{SOL}}$ (and thus against p times SOL) in the ℓ_p^p objective to its regret in the ℓ_p objective.

We first state the technical lemma relating the ℓ_p^p and ℓ_p objectives under the farness condition. Informally, this lemma says that if we want to choose a realization maximizing the regret of ALG against SOL in (an approximation of) the ℓ_p-objective, we should always include a client whose distance to ALG exceeds their distance to SOL by a factor larger than p. This contrasts (and limits) the example given at the beginning of this section, where we showed that including clients whose distance to ALG exceeds the distance to SOL by a smaller factor can actually reduce the regret of ALG against SOL. In turn, if *all* clients are closer to SOL are closer by a factor of p, then the realization that maximizes regret in the ℓ_p-objective is also the realization that maximizes regret in the ℓ_p^p-objective.

LEMMA 3.8. *Suppose ALG and SOL are two solutions to an ℓ_p-clustering instance, such that there is a subset of clients C^* with the following property: for every client in C^*, the connection cost in ALG is greater than p times the connection cost in SOL, while for every client not in C^*, the connection cost in SOL is at least the connection cost in ALG. Then, C^* maximizes the following function:*

$$f(C') := \begin{cases} \frac{\text{ALG}_p^p(C') - \text{SOL}_p^p(C')}{\text{ALG}_p^{p-1}(C')} & \text{ALG}_p^p(C') > 0 \\ 0 & \text{ALG}_p^p(C') = 0 \end{cases}.$$

PROOF. Fix any subset of clients C' that does not include j. Let $\text{ALG}_p^q(C')$ be ALG's ℓ_p^q-objective cost on this subset, $\text{SOL}_p^q(C')$ be SOL's ℓ_p^q-objective on this subset, a be ALG's connection cost for j to the pth power, and s be SOL's connection cost for j to the pth power. To do this, we analyze a continuous extension of f, evaluated at C' plus j in fractional amount x,

$$\tilde{f}_j(C', x) = \frac{\text{ALG}_p^p(C') + ax - \text{SOL}_p^p(C') - sx}{(\text{ALG}_p^p(C') + ax)^{(p-1)/p}}.$$

When $x = 0$, this is $f(C')$ (if $f(C')$ is positive) and when $x = 1$, this is $f(C' \cup \{j\})$ (if $f(C' \cup \{j\})$ is positive). Its derivative with respect to x is as follows:

$$\frac{d}{dx}\tilde{f}_j(C', x) = \frac{(a-s)(\text{ALG}_p^p(C') + ax)^{(p-1)/p} - \frac{a(p-1)}{p} \cdot \frac{\text{ALG}_p^p(C') + ax - \text{SOL}_p^p(C') - sx}{(\text{ALG}_p^p(C') + ax)^{1/p}}}{(\text{ALG}_p^p(C') + ax)^{2(p-1)/p}},$$

which has the same sign as

$$(a-s) - \frac{a(p-1)}{p} \cdot \frac{\text{ALG}_p^p(C') + ax - \text{SOL}_p^p(C') - sx}{\text{ALG}_p^p(C') + ax}.$$

If $\text{ALG}_p^p(C') + ax > \text{SOL}_p^p(C') + sx$, i.e., $\tilde{f}_j(C', x)$ is positive, then $\frac{d}{dx}\tilde{f}_j(C', x)$ is negative if $a \le s$. Consider any C' including a client j not in C^*. Suppose $f(C') > 0$. Then $\tilde{f}_j(C', x)$ has a negative

derivative on $[0,1]$ (since $\tilde{f}_j(C',x)$ starts out positive and is increasing as x goes from 1 to 0, i.e., it stays positive), so $f(C' \setminus \{j\}) = \tilde{f}_j(C',0) > \tilde{f}_j(C',1) = f(C')$, and C' cannot be a maximizer of f. If otherwise $f(C') = 0$, then C' clearly cannot be a maximizer of f unless C^* is as well.

Similarly, observe that

$$(a-s) - \frac{a(p-1)}{p} \cdot \frac{\textsc{alg}_p^p(C') + ax - \textsc{sol}_p^p(C') - sx}{\textsc{alg}_p^p(C') + ax} > (a-s) - \frac{a(p-1)}{p} = \frac{a}{p} - s.$$

So $\frac{d}{dx}\tilde{f}_j(C',x)$ is positive if $a > ps$, which holds for all $j \in C^*$. Consider any C' not including a client j in C^*. Suppose $f(C') > 0$. Then $\tilde{f}_j(C',x)$ has a positive derivative on $[0,1]$, so $f(C') = \tilde{f}_j(C',0) < \tilde{f}_j(C',1) = f(C' \cup \{j\})$, and C' cannot be a maximizer of f. If otherwise $f(C') = 0$, then C' clearly cannot be a maximizer of f unless C^* is as well. Since we have shown that every $C' \neq C^*$ cannot be a maximizer of f unless C^* is also a maximizer of f, we conclude that C^* maximizes f. □

Intuitively, this lemma connects the ℓ_p and ℓ_p^p objectives as this subset of clients C^* will also be the set that maximizes the ℓ_p^p regret of ALG vs SOL, and $f(C')$ is (within a factor of p) equal to the ℓ_p regret. We use this lemma along with the ℓ_p^p-clustering with discounts approximation in Lemma 3.4 to design the rounding algorithm for universal ℓ_p-clustering. As in the rounding algorithm for universal k-median, let SOL denote a (virtual) solution whose connection costs are 3 times that of the fractional solution FRAC for all clients. The rounding algorithm solves an ℓ_p^p-clustering with discounts instance, where the discounts are 2 times SOL's connection costs. (Recall that in k-median, the discount was equal to SOL's connection cost. Now, we need the additional factor of 2 for technical reasons.) Let ALG be the solution output by the algorithm of Lemma 3.4 for this problem. We prove the following bound for ALG:

Lemma 3.9. *There exists an algorithm that, given any (α,β)-approximate fractional solution FRAC for ℓ_p-clustering, outputs a $(54p\alpha, 54p\beta + 18p^{1/p})$-approximate integral solution.*

Proof. Let SOL denote a (virtual) solution whose connection costs are 3 times that of the fractional solution FRAC for all clients. The rounding algorithm solves an ℓ_p^p-clustering with discounts instance, where the discounts are 2 times SOL's connection costs. Let ALG be the solution output by the algorithm of Lemma 3.4 for this problem. We also consider an additional virtual solution $\widetilde{\text{SOL}}$, whose connection costs are defined as follows: For clients j such that ALG's connection cost is greater than 18 times SOL's but less than $18p$ times SOL's, we multiply SOL's connection costs by p to obtain connection costs in $\widetilde{\text{SOL}}$. For all other clients, the connection cost in $\widetilde{\text{SOL}}$ is the same as that in SOL. Now, ALG and $18 \cdot \widetilde{\text{SOL}}$ satisfy the condition in Lemma 3.8 and $18 \cdot \widetilde{\text{SOL}}$ is a $(54p\alpha, 54p\beta)$-approximation.

Our goal in the rest of the proof is to bound the regret of ALG against (a constant times) $\widetilde{\text{SOL}}$ by (a constant times) the minimum regret MR. Let us denote this regret as follows:

$$\text{REG} := \max_{C' \subseteq C} \left[\textsc{alg}_p(C') - 18 \cdot \widetilde{\textsc{sol}}_p(C')\right].$$

Note that if REG = 0 (it cannot be negative), then for all realizations C', $\textsc{alg}_p(C') \leq 18 \cdot \widetilde{\textsc{sol}}_p(C')$. In that case, the lemma follows immediately. So, we assume that REG > 0.

Let $C_1 = \text{argmax}_{C' \subseteq C}[\textsc{alg}_p(C') - 18 \cdot \widetilde{\textsc{sol}}_p(C')]$, i.e., the realization defining REG that maximizes the regret for the ℓ_p objective. We need to relate REG to the regret in the ℓ_p^p-objective for us to use the approximation guarantees of ℓ_p^p-clustering with discounts from Lemma 3.4. Lemma 3.8 gives us this relation, since it tells us that C_1 is exactly the set of clients for which ALG's closest cluster center is at a distance of more than 18 times that of $\widetilde{\text{SOL}}$'s closest cluster center. But this means that

C_1 also maximizes the regret for the ℓ_p^p objective, i.e., $C_1 = \text{argmax}_{C'}[\text{ALG}_p^p(C') - 18^p \cdot \widetilde{\text{SOL}}_p^p(C')]$. Then, we have the following:

$$\begin{aligned}\text{REG} &= \frac{\max_{C'\subseteq C}\left[\text{ALG}_p^p(C') - 18^p \cdot \widetilde{\text{SOL}}_p^p(C')\right]}{\sum_{j=0}^{p-1}\text{ALG}_p^j(C_1)\cdot\left(18\cdot\widetilde{\text{SOL}}_p(C_1)\right)^{p-1-j}} \leq \frac{\max_{C'\subseteq C}\left[\text{ALG}_p^p(C') - 18^p \cdot \widetilde{\text{SOL}}_p^p(C')\right]}{\text{ALG}_p^{p-1}(C_1)} \\ &\leq \frac{\max_{C'\subseteq C}\left[\text{ALG}_p^p(C') - 18^p \cdot \text{SOL}_p^p(C')\right]}{\text{ALG}_p^{p-1}(C_1)}.\end{aligned}$$

The last inequality follows since connection costs in $\widetilde{\text{SOL}}$ are at least those in SOL. Note that the numerator in this last expression is exactly the value of the objective for the ℓ_p^p-clustering with discounts problem from Lemma 3.4. Using this lemma, we can now bound the numerator by the optimum for this problem, which in turn is bounded by the objective produced by the minimum regret solution MRS for the ℓ_p^p-clustering with discounts instance,

$$\text{REG} \leq \frac{\max_{C'\subseteq C}\left[\text{ALG}_p^p(C') - 18^p \cdot \text{SOL}_p^p(C')\right]}{\text{ALG}_p^{p-1}(C_1)} \leq \frac{2}{3}\cdot 9^p \cdot \frac{\max_{C'\subseteq C}\left[\text{MRS}_p^p(C') - 2^p \cdot \text{SOL}_p^p(C')\right]}{\text{ALG}_p^{p-1}(C_1)}. \quad (6)$$

First, we bound the numerator in the above expression. Let $C_2 := \text{argmax}_{C'\subseteq C}[\text{MRS}_p^p(C') - 2^p \cdot \text{SOL}_p^p(C')]$ be the realization that maximizes this term. We now relate this term to MR (the first step is by factorization and the second step holds because $2\cdot\text{SOL} = 6\cdot\text{FRAC}$ exceeds the optimal integer solution by to the upper bound of 3 on the integrality gap [4]),

$$\begin{aligned}\max_{C'}\left[\text{MRS}_p^p(C') - 2^p\cdot\text{SOL}_p^p(C')\right] &= \left(\text{MRS}_p(C_2) - 2\cdot\text{SOL}_p(C_2)\right) \\ &\quad\cdot\sum_{j=0}^{p-1}\text{MRS}_p^j(C_2)\cdot(2\cdot\text{SOL}_p(C_2))^{p-1-j} \\ &\leq \text{MR}\cdot\sum_{j=0}^{p-1}\text{MRS}_p^j(C_2)\cdot(2\cdot\text{SOL}_p(C_2))^{p-1-j}.\end{aligned}$$

Using the above bound in Equation (6), we get

$$\text{REG} \leq \frac{2}{3}\cdot 9^p\cdot\text{MR}\cdot\frac{\sum_{j=0}^{p-1}\text{MRS}_p^j(C_2)\cdot(2\cdot\text{SOL}_p(C_2))^{p-1-j}}{\text{ALG}_p^{p-1}(C_1)}. \quad (7)$$

In the rest of proof, we obtain a bound on the last term in Equation (7). We consider two cases. If $\text{ALG}_p(C_1) \geq 9p^{\frac{1}{p}}\cdot\text{MRS}_p(C_2)$ (intuitively, the denominator is large compared to the numerator), then

$$\frac{\sum_{j=0}^{p-1}\text{MRS}_p^j(C_2)2^{p-1-j}\text{SOL}_p^{p-1-j}(C_2)}{\text{ALG}_p^{p-1}(C_1)} \leq p\cdot\frac{\text{MRS}_p^{p-1}(C_2)}{\text{ALG}_p^{p-1}(C_1)} \leq p\cdot(9p^{\frac{1}{p}})^{-p+1} = 9^{-p+1}p^{1/p},$$

The first step uses the fact that $\text{MRS}_p(C_2) \geq 2\text{SOL}_p(C_2)$, so the largest term in the sum is $\text{MRS}_p^{p-1}(C_2)$. Combined with Equation (7), this $\text{REG} \leq 6p^{1/p}\cdot\text{MR}$, giving the lemma statement.

If $\text{ALG}_p(C_1) < 9p^{\frac{1}{p}}\cdot\text{MRS}_p(C_2)$, then we cannot hope to meaningfully bound Equation (7). In this case, however, REG is also bounded by $\text{MRS}_p(C_2)$, which we will eventually bound by MR. More formally, by our assumption that $\text{REG} > 0$, $\text{MRS}_p^p(C_2) - 2^p\cdot\text{SOL}_p^p(C_2) > 0$ and we have $2\text{SOL}_p(C_2) \leq \text{MRS}_p(C_2) \leq \text{SOL}_p(C_2) + \text{MR}$. The first inequality is by our assumption that

$\text{MRS}_p^p(C_2) - 2^p \cdot \text{SOL}_p^p(C_2) > 0$, the second inequality is by definition of MRS, MR and the fact that $\text{SOL}(C_2)$ upper bounds $\text{OPT}(C_2)$. In turn, $\text{SOL}_p(C_2) \leq \text{MR}$, which gives $\text{MRS}_p(C_2) \leq 2\text{MR}$ and thus $\text{REG} \leq \text{ALG}_p(C_1) \leq 18p^{\frac{1}{p}} \cdot \text{MR}$. We note that for all $p \geq 1$, $p^{1/p} \leq e^{1/e} \leq 1.5$. □

We note that the final step requires using discounts equal to twice SOL's connection costs instead of just SOL's connection costs. If we did the latter, then we would have started with the inequality $\text{SOL}_p(C_2) \leq \text{MRS}_p(C_2) \leq \text{SOL}_p(C_2) + \text{MR}$ instead, which does not give us any useful bound on $\text{SOL}(C_2)$ or $\text{MRS}(C_2)$ in terms of just MR. We also note that we chose not to optimize the constants in the final result of Lemma 3.9 in favor of simplifying the presentation.

Theorem 3.1 now follows by using the values of (α, β) from Lemma 3.7 (and a sufficiently small choice of the error parameter ϵ) in the statement of Lemma 3.9 above.

4 UNIVERSAL k-CENTER

In the previous section, we gave universal algorithms for general ℓ_p-clustering problems. Recall that the k-center objective, defined as the maximum distance of a client from its closest cluster center, can also be interpreted as the ℓ_∞-objective in the ℓ_p-clustering framework. Moreover, it is well known that for any n-dimensional vector, its $\ell_{\log n}$ and ℓ_∞ norms differ only a constant factor (see Fact A.1 in Appendix A). Therefore, choosing $p = \log n$ in Theorem 3.3 gives poly-logarithmic approximation bounds for the universal k-center problem. In this section, we give direct techniques that improve these bounds to constants:

THEOREM 4.1. *There exists a* $(3, 3)$*-approximate algorithm for the universal* k*-center problem.*

Recall that F is the set of all cluster centers, so $\min_{i \in F} c_{ij}$ gives the smallest distance from client j to any cluster center that can be opened. In turn, for every client j, its distance to the closest cluster center in the minimum regret solution MRS, $\min_{i \in \text{MRS}} c_{ij}$, must be at most $\text{MR}_j := \min_{i \in F} c_{ij} + \text{MR}$. This is because in the realization where only j appears, the optimal solution has cost $\min_{i \in F} c_{ij}$, and the cost of MRS is just $\min_{i \in \text{MRS}} c_{ij}$. So, we design an algorithm ALG that 3-approximates these distances MR_j, i.e., for every client j, its distance to the closest cluster center in ALG is at most 3MR_j. Indeed, this algorithm satisfies a more general property: Given *any* value r, it produces a set of cluster centers ALG such that every client j is at a distance $\leq 3r_j$ from its closest cluster center (that is, $\min_{i \in \text{ALG}} c_{ij} \leq 3r_j$), where $r_j := \min_{i \in F} c_{ij} + r$. Moreover, if $r \geq \text{MR}$, then the number of cluster centers selected by ALG is at most k (for smaller values of r, ALG might select more than k cluster centers).

Our algorithm ALG is a natural greedy algorithm. We order clients j in increasing order of r_j, and if a client j does not have a cluster center within distance $3r_j$ in the current solution, then we add its closest cluster center in F to the solution.

LEMMA 4.2. *Given a value* r*, the greedy algorithm* ALG *selects cluster centers that satisfy the following properties:*

- *every client* j *is within a distance of* $3r_j = 3(\min_{i \in F} c_{ij} + r)$ *from their closest cluster center.*
- *If* $r \geq$ MR*, then* ALG *does not select more than* k *cluster centers, i.e., the solution produced by* ALG *is feasible for the* k*-center problem.*

PROOF. The first property follows from the definition of ALG.

To show that ALG does not pick more than k cluster centers, we map the cluster center i added in ALG by some client j to its closest cluster center i' in MRS. Now, we claim that no two cluster centers i_1, i_2 in ALG can be mapped to the same cluster center i' in MRS. Clearly, this proves the lemma since MRS has only k cluster centers.

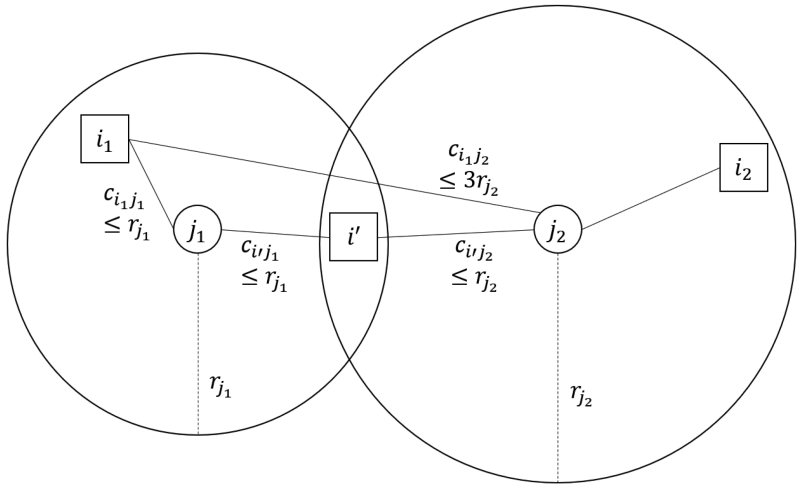

Fig. 2. Two clients j_1, j_2 that are distance at most r_{j_1}, r_{j_2} respectively from the same cluster center i' in MRS cannot cause ALG to add two different cluster centers i_1, i_2.

Suppose i_1, i_2 are two cluster centers in ALG mapped to the same cluster center i' in MRS. Assume without loss of generality that i_1 was added to ALG before i_2. Let j_1, j_2 be the clients that caused i_1, i_2 to be added; since i_2 was added later, we have $r_{j_1} \le r_{j_2}$. The distance from j_2 to i_1 is at most the length of the path (j_2, i', j_1, i_1) (see Figure 2), which is at most $2r_{j_2} + r_{j_1} \le 3r_{j_2}$. But in this case j_2 would not have added a new cluster center i_2, thus arriving at a contradiction. □

We now use the above lemma to prove Theorem 4.1.

Proof of Theorem 4.1. In any realization $C' \subseteq C$, the optimal value of the k-center objective is $\text{OPT}(C') \ge \max_{j \in C'} \min_{i \in F} c_{ij}$, whereas the solution produced by the algorithm ALG given above has objective value at most $3(\max_{j \in C'} \min_{i \in F} c_{ij} + r)$. So, ALG's solution costs at most $3 \cdot \text{OPT}(C') + 3 \cdot r$ for all realizations $C' \subseteq C$. So, if we were able to choose $r = \text{MR}$, then we would prove the theorem. But we do not know the value of MR (in fact, this is **NP**-hard). Instead, we increase the value of r continuously until ALG produces a solution with at most k clients. By Lemma 4.2, we are guaranteed that this will happen for some $r \le \text{MR}$, which then proves the theorem.

Our final observation is that this algorithm can be implemented in polynomial time, since there are only polynomially many possibilities for the k-center objective across all realizations (namely, the set of all cluster center to client distances) and thus polynomially many possible values for MR (the set of all differences between all possible solution costs). So, we only need to run ALG for these values of r in increasing order. □

We note that the greedy algorithm described above can be viewed as an extension of the k-center algorithm in Reference [33] to a $(3, 3)$-approximation for the "k-center with discounts" problem, where the discounts are the minimum distances $\min_{i \in F} c_{ij}$.

5 UNIVERSAL k-MEDIAN WITH FIXED CLIENTS

In this section, we extend the techniques from Section 2 to prove the following theorem:

Theorem 5.1. *If there exists a deterministic polynomial time γ-approximation algorithm for the k-median problem, then for every $\epsilon > 0$ there exists a $(54\gamma + \epsilon, 60)$-approximate universal algorithm for the universal k-median problem with fixed clients.*

By using the derandomized version of the $(2.732+\epsilon)$-approximation algorithm of Li and Svensson [44] for the k-median problem, and appropriate choice of both ϵ parameters, we obtain the following corollary from Theorem 5.1.

Corollary 5.2. *For every $\epsilon > 0$, there exists a $(148+\epsilon, 60)$-approximate universal algorithm for the k-median problem with fixed clients.*

Our high level strategy comprises two steps. In Section 5.2, we show how to find a good fractional solution by approximately solving a linear program. In Section 5.3, we then round the fractional solution in a manner that preserves its regret guarantee within constant factors. As discussed in Section 1.1, for simplicity our algorithm's description and analysis will avoid the notion of demands and instead equivalently view the input as specifying a set of fixed and unfixed clients, of which multiple might exist at the same location.

5.1 Preliminaries

In addition to the preliminaries of Section 2, we will use the following tools:

Submodular Maximization over Independence Systems. An *independence system* comprises a ground set E and a set of subsets (called *independent sets*) $\mathcal{I} \subseteq 2^E$ with the property that if $A \subseteq B$ and $B \in \mathcal{I}$, then $A \in \mathcal{I}$ (the *subset closed* property). An independent set S in $\mathcal{I}$ is *maximal* if there does not exist $S' \supset S$ such that $S' \in \mathcal{I}$. Note that one can define an independence system by specifying the set of maximal independent sets $\mathcal{I}'$ only, since the subset closed property implies $\mathcal{I}$ is simply all subsets of sets in $\mathcal{I}'$. An independence system is a 1-*independence system* (or 1-*system* in short) if all maximal independent sets are of the same size. The following result on maximizing submodular functions over 1-independence systems follows from a more general result given implicitly in Reference [48] and more formally in Reference [14].

Theorem 5.3. *There exists a polynomial time algorithm that given a 1-independence system $(E, \mathcal{I})$ and a non-negative monotone submodular function $f : 2^E \to \mathbb{R}^+$ defined over it, finds a $\frac{1}{2}$-maximizer of f, i.e., finds $S' \in \mathcal{I}$ such that $f(S') \geq \frac{1}{2} \max_{S \in \mathcal{I}} f(S)$.*

The algorithm in the above theorem is the natural greedy algorithm, which starts with $S' = \emptyset$ and repeatedly adds to S' the element u that maximizes $f(S' \cup \{u\})$ while maintaining that $S' \cup \{u\}$ is in $\mathcal{I}$, until no such addition is possible.

Incremental ℓ_p-Clustering. We will also use the *incremental ℓ_p-clustering* problem that is defined as follows: Given an ℓ_p-clustering instance and a subset of the cluster centers S (the "existing" cluster centers), find the minimum cost solution to the ℓ_p-clustering instance with the additional constraint that the solution must contain all cluster centers in S. When $S = \emptyset$, this is just the standard ℓ_p-clustering problem, and this problem is equivalent to the standard ℓ_p-clustering problem by the following lemma:

Lemma 5.4. *If there exists a γ-approximation algorithm for the ℓ_p-clustering problem, then there exists a γ-approximation for the incremental ℓ_p-clustering problem.*

Proof of Lemma 5.4. The γ-approximation for incremental ℓ_p-clustering is as follows: Given an instance I of incremental ℓ_p-clustering with clients C and existing cluster centers S, create a ℓ_p-clustering instance I' that has the same cluster centers and clients as the ℓ_p-clustering instance except that at the location of every cluster center in S, we add a client with demand $\gamma |C|^{1/p} \max_{i,j} c_{ij} + 1$.

Let T^* be the solution that is a superset of S of size k that achieves the lowest cost of all such supersets in instance I. Let T be the output of running a γ-approximation algorithm for ℓ_p-clustering

on I'. Then we wish to show T is a superset of S and has cost at most γ times the cost of T^* in instance I.

Any solution that buys all cluster centers in S has the same cost in I and I'. Then we claim it suffices to show that T is a superset of S. If T is a superset of S, then since both T and T^* are supersets of S and since T is a γ-approximation in instance I', its cost in I' is at most γ times the cost of T^* in I'. This in turn implies T has cost at most γ times the cost of T^* in I, giving the Lemma.

Assume without loss of generality no two cluster centers are distance 0 away from each other. To show that T is a superset of S, note that in instance I' any solution that does not buy a superset of S is thus at least distance 1 from the location of some cluster center in S and thus pays cost at least $\gamma|C|^{1/p}\max_{i,j} c_{ij} + 1$ due to one of the added clients. However, any solution that is a superset of S is distance 0 from all the added clients and thus only has to pay connection costs on clients in C, which in turn means it has cost at most $|C|^{1/p}\max_{i,j} c_{ij}$. Since T is the output of a γ-approximation algorithm, T thus has cost at most $\gamma|C|^{1/p}\max_{i,j} c_{ij}$, which means T must be a superset of S. □

5.2 Obtaining a Fractional Solution for Universal k-Median with Fixed Clients

Let $C_f \subseteq C$ denote the set of fixed clients and for any realization of clients C' satisfying $C_f \subseteq C' \subseteq C$, let $\textsc{opt}(C')$ denote the cost of the optimal solution for C'. The universal k-median LP is given by

$$\begin{aligned}
&&\min\ r \quad &(r \text{ denotes maximum regret across all demand realizations})\\
\text{s.t.}\quad &&\sum_{i\in F} x_i \le k \quad &(x_i = 1 \text{ if we open cluster center } i)\\
&\forall i \in F, j \in C: &y_{ij} \le x_i \quad &(y_{ij} = 1 \text{ if cluster center } i \text{ is serving client } j)\\
&\forall j \in C: &\sum_{i\in F} y_{ij} \ge 1 &\\
&\forall C_f \subseteq C' \subseteq C: &\sum_{j\in C'}\sum_{i\in F} c_{ij}y_{ij} - \textsc{opt}(C') \le r &\qquad (8)\\
&\forall i \in F, j \in C: &x_i, y_{ij} \in [0,1]. &
\end{aligned}$$

Note that Equation (8) and the objective function distinguish this LP from the standard k-median LP. We call Equation (8) the *regret constraint set*. For a fixed fractional solution x, y, r, our goal is to approximately separate the regret constraint set, since all other constraints can be separated exactly. In the rest of this subsection, we describe our approximate separation oracle and give its analysis.

Let $S(C')$ denote the cost of the solution $S \subseteq F$ in realization C' (that is, $S(C') = \sum_{j\in C'} \min_{i\in S} c_{ij}$). Since $\textsc{opt}(C') = \min_{S:S\subseteq F,|S|=k} S(C')$, separating the regret constraint set exactly is equivalent to deciding if the following holds:

$$\forall S: S \subseteq F, |S| = k: \quad \max_{C': C_f \subseteq C' \subseteq C} \left[\sum_{j\in C'}\sum_{i\in F} c_{ij}y_{ij} - S(C')\right] \le r. \qquad (9)$$

By splitting the terms $\sum_{j\in C'}\sum_{i\in F} c_{ij}y_{ij}$ and $S(C')$ into terms for C_f and $C' \setminus C_f$, we can rewrite Equation (9) as follows:

$$\max_{C_f\subseteq C'\subseteq C, S\subseteq F, |S|=k} \sum_{j\in C'}\sum_{i\in F} c_{ij}y_{ij} - S(C') \le r$$

$$\forall S \subseteq F, |S| = k: \max_{C_f\subseteq C'\subseteq C} \sum_{j\in C'}\left[\sum_{i\in F} c_{ij}y_{ij} - S(C')\right] \le r$$

$$\forall S \subseteq F, |S| = k : \max_{C_f \subseteq C' \subseteq C} \sum_{j \in C' \setminus C_f} \left[\sum_{i \in F} c_{ij} y_{ij} + \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} - S(C' \setminus C_f) - S(C_f) \right] \leq r$$

$$\forall S \subseteq F, |S| = k : \max_{C_f \subseteq C' \subseteq C} \left[\sum_{j \in C' \setminus C_f} \sum_{i \in F} c_{ij} y_{ij} - S(C' \setminus C_f) \right] \leq S(C_f) - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r$$

$$\forall S \subseteq F, |S| = k : \max_{C^* \subseteq C \setminus C_f} \left[\sum_{j \in C^*} \sum_{i \in F} c_{ij} y_{ij} - S(C^*) \right] \leq S(C_f) - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r.$$

For fractional solution y, let

$$f_y(S) = \max_{C^* : C^* \subseteq C \setminus C_f} \left[\sum_{j \in C^*} \sum_{i \in F} c_{ij} y_{ij} - S(C^*) \right]. \tag{10}$$

Note that we can compute $f_y(S)$ for any S easily since the maximizing value of C^* is the set of clients j for which S has connection cost less than $\sum_{i \in F} c_{ij} y_{ij}$. We already know $f_y(S)$ is not submodular. But the term $S(C_f)$ is not fixed with respect to S, so maximizing $f_y(S)$ is not enough to separate Equation (8). To overcome this difficulty, for every possible cost M on the fixed clients, we replace $S(C_f)$ with M and only maximize over solutions S for which $S(C_f) \leq M$ (for convenience, we will call any solution S for which $S(C_f) \leq M$ an M-*cheap* solution):

$$\forall M \in \left\{ 0, 1, \ldots, |C_f| \max_{i,j} c_{ij} \right\} : \max_{S : S \subseteq F, |S| = k, S(C_f) \leq M} f_y(S) \leq M - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r. \tag{11}$$

Note that this set of inequalities is equivalent to Equation (8), but it has the advantage that the left-hand side is approximately maximizable and the right-hand side is fixed. Hence, these inequalities can be approximately separated. However, there are exponentially many inequalities; so, for any fixed $\epsilon > 0$, we relax to the following polynomially large set of inequalities:

$$\forall M \in \left\{ 0, 1, 1+\epsilon, \ldots, (1+\epsilon)^{\lceil \log_{1+\epsilon}(|C_f| \max_{i,j} c_{ij}) \rceil + 1} \right\} :$$

$$\max_{S : S \subseteq F, |S| = k, S(C_f) \leq M} f_y(S) \leq M - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r. \tag{12}$$

Separating inequality Equation (12) for a fixed M corresponds to submodular maximization of $f_y(S)$, but now subject to the constraints $|S| = k$ and $S(C_f) \leq M$ as opposed to just $|S| = k$. Let $\mathcal{S}_M$ be the set of all $S \subseteq F$ such that $|S| = k$ and $S(C_f) \leq M$. Since $f_y(S)$ is monotone, maximizing $f_y(S)$ over $\mathcal{S}_M$ is equivalent to maximizing $f_y(S)$ over the independence system $(F, \mathcal{I}_M)$ with maximal independent sets $\mathcal{S}_M$.

Then all that is needed to approximately separate Equation (12) corresponding to a fixed M is an oracle for deciding membership in $(F, \mathcal{I}_M)$. Recall that $S \subseteq F$ is in $(F, \mathcal{I}_M)$ if there exists a set $S' \supseteq S$ such that $|S'| = k$ and $S'(C_f) \leq M$. But even deciding membership of the empty set in $(F, \mathcal{I}_M)$ requires one to solve a k-median instance on the fixed clients, which is in general NP-hard. More generally, we are required to solve an instance of the incremental k-median problem (see Section 5.1) with existing cluster centers in S.

While exactly solving incremental k-median is NP-hard, we have a constant approximation algorithm for it (call it A), by Lemma 5.4. So, we could define a new system $(F, \mathcal{I}'_M)$ that contains a set $S \subseteq F$ if the output of A for the incremental k-median instance with existing cluster centers S has cost at most M. But $(F, \mathcal{I}'_M)$ may no longer be a 1-system, or even an independence system. To restore the subset closed property, the membership oracle needs to ensure that (a) if a subset

GreedyMax((x, y, r), F, C_f, C, M, T_0, f):
Input: Fractional solution (x, y, r), set of cluster centers F, set of fixed clients C_f, set of all clients C, value M, M-cheap solution T_0, submodular objective $f : 2^F \to \mathbb{R}^+$

$S_0 \leftarrow \emptyset$
$F_0 \leftarrow F$
for l from 1 to k **do**
  for Each cluster center i in $F_{l-1} \setminus S_{l-1}$ **do**
    if $S_{l-1} \cup \{i\} \subseteq T_0$ and T_0 is M-cheap **then**
      $T_{l,i} \leftarrow T_0$
    else
      if For some l', i', $S_{l-1} \cup \{i\} \subseteq T_{l',i'}$ and $T_{l',i'}$ is M-cheap **then**
        $T_{l,i} \leftarrow T_{l',i'}$
      else
        $T_{l,i} \leftarrow$ Output of γ-approximation algorithm on incremental ℓ_p-clustering instance with cluster centers F_{l-1}, existing cluster centers $S_{l-1} \cup \{i\}$ and clients C_f.
      end if
    end if
  end for
  $F_l \leftarrow F_{l-1}$
  for Each cluster center i in $F_l \setminus S_{l-1}$ **do**
    if i does not appear in any $T_{l,i'}$ that is M-cheap **then**
      $F_l \leftarrow F_l \setminus \{i\}$
    end if
  end for
  $S_l \leftarrow S_{l-1} \cup \{\text{argmax}_{i \in F_l \setminus S_{l-1}} f(S_{l-1} \cup \{i\})\}$
end for
return S_k

Fig. 3. Modified greedy submodular maximization algorithm.

$S' \subseteq S$ is determined to not be in $(F, \mathcal{I}'_M)$, then S is not either, and (b) if a superset $S' \supseteq S$ is determined to be in $(F, \mathcal{I}'_M)$, then so is S.

We now give the modified greedy maximization algorithm GreedyMax that we use to try to separate one of the inequalities in Equation (12), which uses a built-in membership oracle that ensures the above properties hold. Pseudocode is given in Figure 3, and we informally describe it here. GreedyMax initializes $S_0 = \emptyset$, $F_0 = F$, and starts with a M-cheap k-median solution T_0 (generated by running a γ-approximation on the k-median instance involving only fixed clients C_f). In iteration l, GreedyMax starts with a partial solution S_{l-1} with $l-1$ cluster centers, and it is considering adding cluster centers in F_{l-1} to S_{l-1}. For each cluster center i in F_{l-1}, GreedyMax generates some k-median solution $T_{l,i}$ containing $S_{l-1} \cup \{i\}$ to determine if $S_{l-1} \cup \{i\}$ is in the independence system. If a previously generated solution, T_0 or $T_{l',i'}$ for any l', i', contains $S_{l-1} \cup \{i\}$ and is M-cheap, then $T_{l,i}$ is set to this solution. Otherwise, GreedyMax runs the incremental k-median approximation algorithm on the instance with existing cluster centers in $S_{l-1} \cup \{i\}$, the only cluster centers in the instance are F_{l-1}, and the client set is C_f. It sets $T_{l,i}$ to the solution generated by the approximation algorithm.

After generating the set of solutions $\{T_{l,i}\}_{i \in F_{l-1}}$, if one of these solutions contains $S_{l-1} \cup \{i\}$ and is M-cheap, then GreedyMax concludes that $S_{l-1} \cup \{i\}$ is in the independence system. This, combined with the fact that these solutions may be copied from previous iterations ensures property

SepOracle$((x, y, r), F, C_f, C)$:
Input: A fractional solution x, y, r, set of cluster centers F, set of fixed clients C_f, set of all clients C

1: **if** Any constraint in the universal k-median LP except the regret constraint set is violated **then return** the violated constraint
2: **end if**
3: $T_0 \leftarrow$ output of γ-approximation algorithm A for k-median run on instance with cluster centers F, clients C_f
4: **for** $M \in \{0, 1, 1+\epsilon, (1+\epsilon)^2, \ldots (1+\epsilon)^{\lceil \log_{1+\epsilon}(\gamma |C_f| \max_{i,j} c_{ij}) \rceil + 1}\}$ such that T_0 is M-cheap **do**
5: $\quad S' \leftarrow$ GreedyMax$((x, y, r), F, C_f, C, M, T_0, f_y)$
6: $\quad C' \leftarrow \text{argmax}_{C^* \subseteq C \setminus C_f} [\sum_{j \in C^*} \sum_{i \in F} c_{ij} y_{ij} - S'(C^*)]$
7: $\quad$ **if** $\sum_{j \in C'} \sum_{i \in F} c_{ij} y_{ij} - S(C') > M - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r$ **then**
8: $\quad\quad$ **return** $\sum_{j \in C'} \sum_{i \in F} c_{ij} y_{ij} - S(C') \leq M - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r$
9: $\quad$ **end if**
10: **end for**
11: **return** "Feasible"

Fig. 4. Approximate separation oracle for universal k-median.

(b) holds (as the M-cheap solutions generated by GreedyMax are implicitly considered to be in the independence system). Otherwise, since GreedyMax was unable to find an M-cheap superset of $S_{l-1} \cup \{i\}$, it considers $S_{l-1} \cup \{i\}$ to not be in the independence system. In accordance with these beliefs, GreedyMax initializes F_l as a copy of F_{l-1}, and then removes any i such that it did not find an M-cheap superset of $S_{l-1} \cup \{i\}$ from F_l and thus from future consideration, ensuring property (a) holds. It then greedily adds to S_{l-1} the i in F_l that maximizes $f_y(S_{l-1} \cup \{i\})$ as defined before to create a new partial solution S_l. After the kth iteration, GreedyMax outputs the solution S_k.

Our approximate separation oracle, SepOracle, can then use GreedyMax as a subroutine. Pseudocode is given in Figure 4, and we give an informal description of the algorithm here. SepOracle checks all constraints except the regret constraint set, and then outputs any violated constraints it finds. If none are found, it then runs a k-median approximation algorithm on the instance containing only the fixed clients to generate a solution T_0. For each M that is 0 or a power of $1 + \epsilon$ (as in Equation (12)), if T_0 is M-cheap, it then invokes GreedyMax for this value of M (otherwise, GreedyMax will consider the corresponding independence system to be empty, so there is no point in running it), passing T_0 to GreedyMax. It then checks the inequality $\sum_{j \in C'} \sum_i c_{ij} y_{ij} - S(C') \leq M - \sum_{j \in C_f} \sum_i c_{ij} y_{ij} + r$ for the solution S outputted by GreedyMax, and outputs this inequality if it is violated.

This completes the intuition behind and description of the separation oracle. We now move on to its analysis. First, we show that GreedyMax always finds a valid solution.

Lemma 5.5. *GreedyMax always outputs a set S_k of size k when called by SepOracle.*

Proof. Note that GreedyMax is only invoked if T_0 is M-cheap. This implies some $T_{1,i}$ is M-cheap since some $T_{1,i}$ will be initialized to T_0. Then, it suffices to show that in the lth iteration, there is some i that can be added to S_l. If this is true, then it implies S_k is of size k since k elements are added across all k iterations.

This is true in iteration 1 because some $T_{1,i}$ is M-cheap and thus any element of $T_{1,i}$ is in F_1 and can be added. Assume this is inductively true in iteration l, i.e., i is added to S_l in iteration l because i is in some M-cheap $T_{l,i'}$. Since $T_{l,i'}$ is M-cheap, no element of $T_{l,i'}$ is deleted from F_l.

Then in iteration $l+1$, for all i'' in $T_{l,i'} \setminus S_l$ (a set of size $k-l$, i.e., non-empty), $T_{l+1,i''}$ can be initialized to $T_{l,i'}$. Then all such i'' can be added to S_{l+1} because all such i'' satisfy that $T_{l+1,i''}$ is M-cheap and thus are in F_{l+1}. By induction, in all iterations, there is some i that can be added to S_l, giving the Lemma. □

Then, the following lemma asserts that GreedyMax is indeed performing greedy submodular maximization over some 1-system.

Lemma 5.6. *Fix any run of GreedyMax. Consider the values of $S_l, T_{l,i}, F_l$ for all l, i (defined as in Figure 3) at the end of this run. Let $\mathcal{B}$ be the set containing $S_{l-1} \cup \{i\}$ for each l, $i \notin F_l$, $i \notin S_{l-1}$. Let $(F, \mathcal{S})$ be the independence system for which the set of maximal independent sets $\mathcal{S}_{max}$ consists of all size k subsets S of F such that no subset of S is in $\mathcal{B}$, and S is M-cheap. Then the following properties hold:*

(1) *For any l and any $i \notin F_l$, $S_{l-1} \cup \{i\}$ is not in $\mathcal{S}$*
(2) *For any l and any $i \in F_l$, $S_{l-1} \cup \{i\}$ is in $\mathcal{S}$*
(3) *$(F, \mathcal{S})$ is a 1-system*

Proof.

Property 1. *This property is immediate from the definition of $\mathcal{B}$ and $\mathcal{S}$.*

Property 2. *Fix any $l, i \in F_l$. We want to show that $S_{l-1} \cup \{i\}$ is in $\mathcal{S}$. Since $i \in F_l$, there exists some $T_{l,i'}$ such that $S_{l-1} \cup \{i\}$ is a subset of $T_{l,i'}$ and $T_{l,i'}$ is M-cheap (otherwise, i would have been deleted from F_l). If we can show that $T_{l,i'}$ is in $\mathcal{S}_{max}$, then we immediately get that $S_{l-1} \cup \{i\}$ is in $\mathcal{S}$.*

Suppose not. Since $T_{l,i'}$ is M-cheap, this must be because some subset of $T_{l,i'}$ is of the form $S_{l'-1} \cup \{i''\}$ for $i'' \notin F_{l'}, i'' \notin S_{l'-1}$. In particular, consider the smallest value l' for which this is true, i.e., let l' be the iteration in which i'' was deleted from $F_{l'}$.

If $l' < l$, since i'' was deleted from $F_{l'}$, then i'' cannot appear in any M-cheap solution containing $S_{l'-1}$ generated by the incremental k-median approximation algorithm before the end of iteration l' (otherwise, $T_{l',i''}$ could be initialized to this solution, preventing i'' from being deleted). Since i'' is not in $F_{l'}$ (and thus not in $F_{l'+1} \ldots F_l$), in iterations $l'+1$ to l the approximation algorithm is not allowed to use i''. So no M-cheap solution is ever generated by the approximation algorithm that is a superset of $S_{l'-1} \cup \{i''\}$. But $T_{l,i'}$ is a M-cheap superset of $S_{l'-1} \cup \{i''\}$ that must have been generated by the approximation algorithm at some point, a contradiction.

Thus we can assume $l' \geq l$. However, recall that $T_{l,i'}$ is an M-cheap solution containing $S_{l'} \cup \{i''\}$. If $l' = l$, then this prevents i'' from being deleted in iteration l', giving a contradiction. If $l' > l$, then $T_{l',i''}$ can be initialized to a M-cheap superset of $S_{l'} \cup \{i''\}$, since $T_{l,i'}$ is such a superset. This also prevents i'' from being deleted in iteration l'', giving a contradiction.

In all cases assuming $T_{l,i'}$ is not in $\mathcal{S}_{max}$ leads to a contradiction, so $T_{l,i'}$ is in $\mathcal{S}_{max}$ and thus $S_{l-1} \cup \{i\}$ is in $\mathcal{S}$.

Property 3. *$\mathcal{S}$ is defined so that all maximal independent sets are of size k, giving the property.*

□

Corollary 5.7. *Any run of GreedyMax outputs an M-cheap, $\frac{1}{2}$-maximizer of $f_y(S)$ over the system $(F, \mathcal{S})$ as defined in Lemma 5.6.*

Proof. Properties 1 and 2 in Lemma 5.6 imply that at each step, GreedyMax adds the element to its current solution that maximizes the objective $f_y(S)$ while maintaining that the current solution is in $\mathcal{S}$. Thus GreedyMax is exactly the greedy algorithm in Theorem 5.3 for maximizing a monotone submodular objective over an independence system. By Lemma 5.5, GreedyMax always finds

a maximal independent set, and the definition of $\mathcal{S}$ guarantees that this maximal independent set is M-cheap. Lemma 2.4 gives that $f_y(S)$ is a monotone submodular function of S. Then, Property 3 combined with Theorem 5.3 implies the solution output by GreedyMax is a $\frac{1}{2}$-maximizer. □

Of course, maximizing over an arbitrary 1-system is of little use. In particular, we would like to show that the 1-system Lemma 5.6 shows we are maximizing over approximates the 1-system of subsets of solutions whose cost on the fixed clients is at most M. The next lemma shows that while all such solutions may not be in this 1-system, all solutions that are $\frac{M}{\gamma}$-cheap are.

Lemma 5.8. *In any run of GreedyMax, let $\mathcal{S}$ be defined as in Lemma 5.6. For the value M passed to GreedyMax in this run and any solution S that is $\frac{M}{\gamma}$-cheap, $S \in \mathcal{S}$.*

Proof. Fix any such S. Let $\mathcal{B}$ be defined as in Lemma 5.6. For any element B of $\mathcal{B}$, it must be the case that running a γ-approximation on the incremental k-median instance with existing cluster centers B produced a solution with cost greater than M. This implies that for any B in $\mathcal{B}$, the incremental k-median instance with existing cluster centers B has optimal solution with cost greater than $\frac{M}{\gamma}$. However, for any subset S' of S, the optimal solution to the incremental k-median instance with existing cluster centers S' has cost at most $\frac{M}{\gamma}$ since S is a feasible solution to this instance that is $\frac{M}{\gamma}$-cheap. Thus no subset of S is in $\mathcal{B}$, and hence S is in $\mathcal{S}$. □

Last, we show that SepOracle never incorrectly outputs that a point is infeasible, i.e., that the region SepOracle considers feasible strictly contains the region that is actually feasible in the universal k-median LP.

Lemma 5.9. *If x, y, r is feasible for the universal k-median LP, then SepOracle outputs "Feasible."*

Proof. SepOracle can exactly check all constraints besides the regret constraint set, so assume that if SepOracle outputs that x, y, r is not feasible, then it outputs that $\sum_{j\in C'} \sum_{i\in F} c_{ij}y_{ij} - S(C') \leq M - \sum_{j\in C_f} \sum_{i\in F} c_{ij}y_{ij} + r$ is violated for some M, S. In particular, it only outputs that this constraint is violated if it actually is violated. If this constraint is violated, then since by Corollary 5.7 S is M-cheap,

$$\sum_{j\in C'} \sum_{i\in F} c_{ij}y_{ij} - S(C') > M - \sum_{j\in C_f} \sum_{i\in F} c_{ij}y_{ij} + r$$

$$\sum_{j\in C'} \sum_{i\in F} c_{ij}y_{ij} + \sum_{j\in C_f} \sum_{i\in F} c_{ij}y_{ij} > S(C') + M + r$$

$$\sum_{j\in C'\cup C_f} \sum_{i\in F} c_{ij}y_{ij} > S(C') + M + r$$

$$\geq S(C') + S(C_f) + r = S(C' \cup C_f) + r \geq \text{OPT}(C' \cup C_f) + r,$$

which implies the point x, y, r is not feasible for the universal k-median LP. □

We now have all the tools to prove our overall claim:

Lemma 5.10. *If there exists a deterministic polynomial-time γ-approximation algorithm for the k-median problem, then for every $\epsilon > 0$ there exists a deterministic algorithm that outputs a $(2\gamma(1+\epsilon), 2)$-approximate fractional solution to the universal k-median problem in polynomial time.*

Proof. We use the ellipsoid method where SepOracle is used as the separation oracle. By Lemma 5.9 since the minimum regret solution is a feasible solution to the universal k-median LP, it is also considered feasible by SepOracle. Then, the solution x^*, y^*, r^* output by the ellipsoid method satisfies $r^* \leq \text{MR}$.

Suppose the ellipsoid method outputs x^*, y^*, r^* such that x^*, y^* are not a $(2\gamma(1+\epsilon), 2)$-approximate solution. This means there exists $S, C_f \subseteq C' \subseteq C$ such that

$$\begin{aligned}\sum_{j\in C'}\sum_{i\in F} c_{ij}y^*_{ij} &> 2\gamma(1+\epsilon)S(C') + 2\cdot \text{MR}\\ \sum_{j\in C'\setminus C_f}\sum_{i\in F} c_{ij}y^*_{ij} - S(C'\setminus C_f) &> 2\gamma(1+\epsilon)S(C_f) + (2\gamma(1+\epsilon)-1)S(C'\setminus C_f)\\ &\quad - \sum_{j\in C_f}\sum_{i\in F} c_{ij}y^*_{ij} + 2\cdot \text{MR}\\ &\geq 2\left[\gamma(1+\epsilon)S(C_f) - \sum_{j\in C_f}\sum_{i\in F} c_{ij}y^*_{ij} + \text{MR}\right].\end{aligned}$$

Thus, for the value of M in the set $\{0, 1, 1+\epsilon, (1+\epsilon)^2, \ldots (1+\epsilon)^{\lceil \log_{1+\epsilon}(\gamma |C_f| \max_{i,j} c_{ij})\rceil + 1}\}$ contained in the interval $[\gamma S(C_f), \gamma(1+\epsilon)S(C_f)]$, we have

$$\sum_{j\in C'\setminus C_f}\sum_{i\in F} c_{ij}y^*_{ij} - S(C'\setminus C_f) \geq 2\left[M - \sum_{j\in C_f}\sum_{i\in F} c_{ij}y^*_{ij} + \text{MR}\right] \geq 2\left[M - \sum_{j\in C_f}\sum_{i\in F} c_{ij}y^*_{ij} + r^*\right].$$

The last inequality follows since $r^* \leq \text{MR}$. Then, consider the iteration in SepOracle where it runs GreedyMax for this value of M. Since $M \geq \gamma S(C_f)$, S is $\frac{M}{\gamma}$-cheap. Thus by Lemma 5.8, S is part of the independence system $\mathcal{S}$ specified in Lemma 5.6 that GreedyMax finds a maximizer for in this iteration, and thus the maximum of the objective in this independence system is at least $2[M - \sum_{j\in C_f}\sum_i c_{ij}y^*_{ij} + r^*]$. By Corollary 5.7, SepOracle thus finds some $S', C'' \subseteq C \setminus C_f$ such that S' is M-cheap and for which $\sum_{j\in C''}\sum_i c_{ij}y^*_{ij} - S'(C'')$ is at least $M - \sum_{j\in C_f}\sum_i c_{ij}y^*_{ij} + r^*$. But this means SepOracle will output that x^*, y^*, r^* is infeasible, which means the ellipsoid algorithm cannot output this solution, a contradiction. □

5.3 Rounding the Fractional Solution for Universal k-Median with Fixed Clients

Proof of Theorem 5.1. The algorithm is as follows: Use the algorithm of Lemma 5.10 with error parameter $\frac{\epsilon}{54\gamma}$ to find a $(2\gamma(1+\frac{\epsilon}{54\gamma}), 2)$-approximate fractional solution. Let f_j be the connection cost of this fractional solution for client j. Construct a k-median with discounts instance with the same clients C and cluster centers F where client j has discount 0 if it was originally a fixed client, and discount $3f_j$ if it was originally a unfixed client. The solution to this instance given by Lemma 2.3 is the solution for the universal k-median instance.

Again using the integrality gap upper bound of 3 for k-median, we have the following:

$$\text{MR} = \max_{C'}[\text{MRS}(C') - \text{OPT}(C')] \geq \max_{C'}\left[\text{MRS}(C') - 3\sum_{j\in C'} f_j\right] = \sum_{j\in C_f}(m_j - 3f_j) + \sum_{j\in C\setminus C_f}(m_j - 3f_j)^+. \tag{13}$$

The cost of the minimum regret solution in the k-median with discounts instance is given by

$$\begin{aligned}\sum_{j\in C_f} m_j + \sum_{j\in C\setminus C_f}(m_j - 3f_j)^+ &= \sum_{j\in C_f} 3f_j + \sum_{j\in C_f}(m_j - 3f_j) + \sum_{j\in C\setminus C_f}(m_j - 3f_j)^+\\ &\leq \sum_{j\in C_f} 3f_j + \text{MR}, \text{ by Equation (13).}\end{aligned} \tag{14}$$

Let c_j be the connection cost of the algorithm's solution for client j. Lemma 2.3 and Equation (14) give

$$\begin{aligned}
&\sum_{j\in C_f} c_j + \sum_{j\in C\setminus C_f} (c_j - 9\cdot 3f_j)^+ \le 6\left[\sum_{j\in C_f} 3f_j + \text{MR}\right]\\
&\sum_{j\in C_f} c_j + \sum_{j\in C\setminus C_f} (c_j - 27f_j)^+ \le \sum_{j\in C_f} 18f_j + 6\cdot \text{MR}\\
\implies \quad &\max_{C'}\left[\sum_{j\in C'} c_j - 27\sum_{j\in C'} f_j\right] = \sum_{j\in C_f}(c_j - 27f_j) + \sum_{j\in C\setminus C_f}(c_j - 27f_j)^+ \le 6\cdot \text{MR}.
\end{aligned} \tag{15}$$

Lemma 5.10 then gives that for any valid C',

$$\text{FRAC}(C') = \sum_{j\in C'} f_j \le 2\gamma\left(1 + \frac{\epsilon}{54\gamma}\right)\cdot \text{OPT}(C') + 2\cdot \text{MR}. \tag{16}$$

Using Equations (15) and (16), we can then conclude that

$$\forall C_f \subseteq C' \subseteq C : \sum_{j\in C'} c_j \le 27\sum_{j\in C'} f_j + 6\cdot \text{MR} \le (54\gamma+\epsilon)\cdot \text{OPT}(C') + 60\cdot \text{MR}. \quad \square$$

6 UNIVERSAL ℓ_p-CLUSTERING WITH FIXED CLIENTS

In this section, we give the following theorem:

THEOREM 6.1. *For all $p \ge 1$, if there exists a γ-approximation for ℓ_p-clustering, then for all $\epsilon > 0$ there exists a $(54p\gamma\cdot 2^{1/p} + \epsilon, 108p^2 + 6p^{1/p} + \epsilon)$-approximate universal algorithm for ℓ_p-clustering with fixed clients.*

In particular, we get from known results [1, 30]:

- A $(162p^2\cdot 2^{1/p} + \epsilon, 108p^2 + 18p^{1/p} + \epsilon)$-approximate universal algorithm for ℓ_p-clustering with fixed clients for all $\epsilon > 0$, $p \ge 1$.
- A $(459, 458)$-approximate universal algorithm for k-means with fixed clients.

The algorithm for universal ℓ_p-clustering with fixed clients follows by combining techniques from ℓ_p-clustering and k-median with fixed clients.

6.1 Finding a Fractional Solution

We reuse the subroutine GREEDYMAX to do submodular maximization over an independence system whose bases are M-cheap solutions (that is, solutions with ℓ_p-objective at most M on only the fixed clients), and use the submodular function $f_{y,Y}$ with varying choices of Y as we did for ℓ_p-clustering. We can extend Lemma 3.5 as follows:

LEMMA 6.2. *For any two solutions y, S, if the global maximum of $\text{FRAC}_p(\mathbf{I}) - S_p(\mathbf{I})$ over $1^{C_f}\times[0,1]^{C\setminus C_f}$ is positive, then there is a maximizer that is in $1^{C_f}\times\{0,1\}^{C\setminus C_f}$, i.e.,*

$$\max_{\mathbf{I}\in 1^{C_f}\times[0,1]^{C\setminus C_f}:}\left[\text{FRAC}_p(\mathbf{I}) - S_p(\mathbf{I})\right] = \max_{C_s\subseteq C'\subseteq C}\left[\text{FRAC}_p(C') - S_p(C')\right].$$

The proof follows exactly the same way as Lemma 3.5. In that proof, the property we use of having no fixed clients is that if the global maximum is not the all zeroes vector, then it is positive and so $\text{FRAC}_p(\mathbf{I}) > S_p(\mathbf{I})$. In the statement of Lemma 6.2, we just assume positivity instead. This shows that it is still fine to output separating hyperplanes based on fractional realizations of clients in the

FIXED-ℓ_p-SEPORACLE((x, y, r), F, C_f, C):
Input: A fractional solution x, y, r, set of cluster centers F, set of fixed clients C_f, set of all clients C

1: **if** Any constraint in the universal ℓ_p-clustering LP except the regret constraint set is violated **then return** the violated constraint
2: **end if**
3: $T_0 \leftarrow$ output of γ-approximation algorithm A for ℓ_p-clustering run on instance with cluster centers F, clients C_f
4: $c_{\min} \leftarrow \min_{i\in F, j\in C\setminus C_f} c_{ij}^p$, $c_{\max} \leftarrow \sum_{j\in C\setminus C_f} \max_{i\in F} c_{ij}^p$
5: $Y_f \leftarrow \sum_{j\in C_f}\sum_{i\in F} c_{ij}^p y_{ij}$
6: **for** $M \in \{0, 1, 1+\epsilon, (1+\epsilon)^2, \dots (1+\epsilon)^{\lceil \log_{1+\epsilon}(\gamma |C_f|^{1/p} \max_{i,j} c_{ij})\rceil + 1}\}$ such that T_0 is M-cheap **do**
7: **for** $Y \in \{0, c_{\min}, c_{\min}(1+\epsilon'), c_{\min}(1+\epsilon')^2, \dots c_{\min}(1+\epsilon')^{\lceil \log_{1+\epsilon'} c_{\max}/c_{\min}\rceil}\}$ **do**
8: $S' \leftarrow$ GREEDYMAX((x, y, r), F, C_f, C, M, T_0, $f_{y,Y}$)
9: $\mathbf{I}' \leftarrow \operatorname{argmax}_{\mathbf{I}\in 1^{C_f}\times[0,1]^{C\setminus C_f}: \sum_{j\in C\setminus C_f} d_j \sum_{i\in F} c_{ij}^p y_{ij} \le Y} \sum_{j\in C\setminus C_f} d_j \sum_{i\in F} c_{ij}^p y_{ij} - \sum_{j\in C\setminus C_f} d_j \min_{i\in S} c_{ij}^p$
10: **if** $\frac{1}{p(Y_f+Y)^{1-1/p}}\left[\sum_{j\in C} d_j' \sum_{i\in F} c_{ij}^p y_{ij} - \sum_{j\in C} d_j' \min_{i\in S} c_{ij}^p\right] > r$ **then**
11: **return** $\frac{1}{p(Y_f+Y)^{1-1/p}}\left[\sum_{j\in C} d_j' \sum_{i\in F} c_{ij}^p y_{ij} - \sum_{j\in C} d_j' \min_{i\in S} c_{ij}^p\right] \le r$
12: **end if**
13: **end for**
14: **end for**
15: **return** "Feasible"

Fig. 5. Approximate separation oracle for universal ℓ_p-clustering with fixed clients. GREEDYMAX is the same algorithm as presented in Figure 3 for k-median.

presence of fixed clients. The only time it is maybe not fine is in a fractional realization where if the "regret" of FRAC is negative, but in this case we will not output a separating hyperplane anyway.

LEMMA 6.3. *If there exists a γ-approximation for ℓ_p-clustering, then for all $\epsilon > 0$, $\alpha = 2^{1/p}\gamma(1+\epsilon), \beta = 2p(1+\epsilon)$ there exists an algorithm that outputs an (α, β)-approximate universal fractional solution for ℓ_p-clustering with fixed clients.*

PROOF. If FIXED-ℓ_p-SEPORACLE (given in Figure 5) ever outputs an inequality in the regret constraint set, for the corresponding $Y_f, Y, \mathbf{I}', S$, then let $\text{FRAC}_p^q(\mathbf{I}), \text{SOL}_p^q(\mathbf{I})$ denote the ℓ_p^q costs of the fractional solution and S as before. Then we have by definition of Y_f and the constraint that $\sum_{j\in C} d_j \sum_{i\in F} c_{ij}^p y_{ij} \le Y$:

$$r < \frac{1}{p(Y_f+Y)^{1-1/p}}\left[\sum_{j\in C} d_j' \sum_{i\in F} c_{ij}^p y_{ij} - \sum_{j\in C} d_j' \min_{i\in S} c_{ij}^p\right]$$
$$\le \frac{1}{\sum_{j=0}^{p-1} \text{FRAC}_p^j(\mathbf{I}')\text{SOL}_p^{p-1-j}(\mathbf{I}')}\left[\text{FRAC}_p^p(\mathbf{I}') - \text{SOL}_p^p(\mathbf{I}')\right] = \text{FRAC}_p(\mathbf{I}') - \text{SOL}_p(\mathbf{I}').$$

The second inequality uses that FIXED-ℓ_p-SEPORACLE only outputs an inequality in the regret constraint set such that $\text{FRAC}_p(\mathbf{I}') > \text{SOL}_p(\mathbf{I}')$. We then have by Lemma 6.2 that for any feasible fractional solution, the inequality output by FIXED-ℓ_p-SEPORACLE is satisfied.

Now, suppose there exists some $\mathbf{I}$, SOL such that $\text{FRAC}_p(\mathbf{I}) > \alpha\text{SOL}_p(\mathbf{I}) + \beta r$ (for $r \ge 0$). Consider the values of Y, M iterated over by FIXED-ℓ_p-SEPORACLE such that $\sum_{j\in C\setminus C_f} d_j \sum_{i\in F} c_{ij}^p y_{ij} \le Y \le$

$(1+\epsilon)(\sum_{j\in C\setminus C_f} d_j \sum_{i\in F} c_{ij}^p y_{ij})$ and $\gamma \textsc{sol}_p(C_f) \le M \le \gamma(1+\epsilon)\textsc{sol}_p(C_f)$. Then,

$$\left(\sum_{j\in C'}\sum_{i\in F} c_{ij}^p y_{ij}\right)^{1/p} > \alpha \textsc{sol}_p(C') + \beta r$$

$$\left(\sum_{j\in C'}\sum_{i\in F} c_{ij}^p y_{ij}\right)^{1/p} - \alpha \textsc{sol}_p(C') > \beta r$$

$$\frac{\sum_{j\in C'}\sum_{i\in F} c_{ij}^p y_{ij} - \alpha^p \textsc{sol}_p^p(C')}{\left(\sum_{j\in C'}\sum_{i\in F} c_{ij}^p y_{ij}\right)^{1-1/p}} > \beta r \tag{i}$$

$$\frac{(1+\epsilon)\left(\sum_{j\in C'}\sum_{i\in F} c_{ij}^p y_{ij} - \alpha^p \textsc{sol}_p^p(C')\right)}{\left(Y_f + Y\right)^{1-1/p}} > \beta r \tag{ii}$$

$$\sum_{j\in C'}\sum_{i\in F} c_{ij}^p y_{ij} - \alpha^p \textsc{sol}_p^p(C') > \frac{\beta r \left(Y_f + Y\right)^{1-1/p}}{1+\epsilon}$$

$$\sum_{j\in C'\setminus C_f}\sum_{i\in F} c_{ij}^p y_{ij} - \alpha^p \textsc{sol}_p^p(C'\setminus C_f) > \frac{\beta r (Y_f+Y)^{1-1/p}}{1+\epsilon} + \alpha^p \textsc{sol}_p^p(C_f) - \sum_{j\in C_f}\sum_{i\in F} c_{ij}^p y_{ij}$$

$$\sum_{j\in C'\setminus C_f}\sum_{i\in F} c_{ij}^p y_{ij} - \alpha^p \textsc{sol}_p^p(C'\setminus C_f) > \frac{\beta r (Y_f+Y)^{1-1/p}}{1+\epsilon} + \alpha^p \left(\frac{M}{\gamma(1+\epsilon)}\right)^p - \sum_{j\in C_f}\sum_{i\in F} c_{ij}^p y_{ij}. \tag{iii}$$

(i) follows from the fact that $\textsc{sol}_p(C') > \sum_{j\in C'}\sum_{i\in F} c_{ij}^p$ if $a > b$. (ii) follows from definitions of Y_f, Y. (iii) follows from the choice of M. Let $\mathbf{I}'_{C'}$ denote the vector whose jth element is $\mathbf{I}'_j$ if $j \in C'$ and 0 otherwise. By the analysis in Section 5.1, since SOL is M/γ-cheap it is in the independence system that GreedyMax finds a 1/2-maximizer for. That is, GreedyMax outputs some S and Fixed-ℓ_p-Oracle finds some $\mathbf{I}'$ such that S is M-cheap, $\sum_{j\in C\setminus C_f} d'_j \sum_{i\in F} c_{ij}^p y_{ij} \le Y$, and such that

$$\sum_{j\in C'\setminus C_f} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - \alpha^p S_p^p(\mathbf{I}'_{C\setminus C_f}) > \frac{1}{2}\left[\frac{\beta r (Y_f+Y)^{1-1/p}}{1+\epsilon} + \alpha^p \left(\frac{M}{\gamma(1+\epsilon)}\right)^p - \sum_{j\in C_f}\sum_{i\in F} c_{ij}^p y_{ij}\right]$$

$$\sum_{j\in C'\setminus C_f} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - S_p^p(\mathbf{I}'_{C\setminus C_f}) > \frac{1}{2}\left[\frac{\beta r (Y_f+Y)^{1-1/p}}{1+\epsilon} + \alpha^p \left(\frac{S_p(C_f)}{\gamma(1+\epsilon)}\right)^p\right] - \sum_{j\in C_f} d'_j \sum_{i\in F} c_{ij}^p y_{ij} \tag{iv}$$

$$\sum_{j\in C'\setminus C_f} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - S_p^p(\mathbf{I}'_{C\setminus C_f}) > \frac{\beta r (Y_f+Y)^{1-1/p}}{2(1+\epsilon)} + S_p^p(C_f) - \sum_{j\in C_f} d'_j \sum_{i\in F} c_{ij}^p y_{ij} \tag{v}$$

$$\sum_{j\in C'} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - S_p^p(\mathbf{I}') > \frac{\beta r (Y_f+Y)^{1-1/p}}{2(1+\epsilon)}$$

$$\frac{\sum_{j\in C'} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - S_p^p(\mathbf{I}')}{p(Y_f+Y)^{1-1/p}} > \frac{\beta r}{2p(1+\epsilon)}$$

$$\frac{\sum_{j\in C'} d'_j \sum_{i\in F} c_{ij}^p y_{ij} - S_p^p(\mathbf{I}')}{p(Y_f+Y)^{1-1/p}} > r. \tag{vi}$$

(iv) follows the M-cheapness of S. (v) follows from the choice of α. (vi) follows from the choice of β. So, Fixed-ℓ_p-SepOracle outputs an inequality as desired. □

6.2 Rounding the Fractional Solution

Again, we show how to generalize the approach for rounding fractional solutions for k-median with fixed clients to round fractional solutions for ℓ_p clustering with fixed clients. We extend Lemma 3.8 as follows:

Lemma 6.4. *Suppose ALG and SOL are two (possibly virtual) solutions to an ℓ_p-clustering instance with fixed clients C_f, such that there is a subset of clients $C^* \subset (C \setminus C_f)$ such that for every client in C^* ALG's connection cost is greater than p times SOL's connection cost, and for every client in $C \setminus C_f \setminus C^*$, SOL's connection cost is at least ALG's connection cost. Then*

$$f(C') := \begin{cases} \frac{ALG_p^p(C') - SOL_p^p(C')}{ALG_p^{p-1}(C')} & ALG_p^p(C') > 0 \\ 0 & ALG_p^p(C') = 0 \end{cases}$$

is maximized by $C_f \cup C^*$.

The proof follows exactly as that of Lemma 3.8.

Lemma 6.5. *There exists an algorithm that given any (α, β)-approximate universal fractional solution for ℓ_p-clustering with fixed clients, outputs a $(54p\alpha, 54p\beta + 18p^{1/p})$-approximate universal integral solution.*

Proof. Let SOL be the virtual solution whose connection costs are 3 times the fractional solution's for all clients. The algorithm is to solve the ℓ_p^p-clustering with discounts instance using Lemma 3.4 where the discounts are 0 for fixed clients and 2 times SOL's connection costs for the remaining clients. Note that using these discounts, the ℓ_p^p-clustering with discounts objective equals $\max_{C_f \subseteq C' \subseteq C}[\text{ALG}_p^p(C') - 2^p \cdot \text{SOL}_p^p(C' \setminus C_f)]$ instead of $\max_{C_f \subseteq C' \subseteq C}[\text{ALG}_p^p(C') - 2^p \cdot \text{SOL}_p^p(C')]$.

Let ALG be the output solution. We will again bound ALG's cost against the virtual solution $\widetilde{\text{SOL}}$ whose connection costs are SOL's connection costs times p for non-fixed clients j such that ALG's connection cost to j is at least 18 times SOL's but less than $18p$ times $18 \cdot$ SOL's, and the same as SOL's for the remaining clients.

We use $\max_{C'}$ to denote $\max_{C_f \subseteq C' \subseteq C}$. If $\max_{C'}[\text{ALG}_p(C') - 18\widetilde{\text{SOL}}(C')] \le 0$, then ALG's cost is always bounded by 18 times $\widetilde{\text{SOL}}$'s cost and we are done. So assume $\max'_C[\text{ALG}_p(C') - 18\widetilde{\text{SOL}}_p(C')] > 0$. Let $C_1 = \text{argmax}_{C'}[\text{ALG}_p(C') - 18\widetilde{\text{SOL}}_p(C')]$ and $C_2 = \text{argmax}_{C'}[\text{MRS}_p^p(C') - 2^p \cdot \text{SOL}_p^p(C')]$. Like in the proof of Lemma 3.9, via Lemma 6.4 we have

$$\begin{aligned} \max_{C'} \left[\text{ALG}_p(C') - 18\widetilde{\text{SOL}}_p(C')\right] &= \frac{\max_{C'} \left[\text{ALG}_p^p(C') - 18^p \text{SOL}_p^p(C')\right]}{\text{ALG}_p^{p-1}(C_1)} \\ &= \frac{\max_{C'} \left[\text{ALG}_p^p(C') - 18^p \text{SOL}_p^p(C' \setminus C_f)\right] - 18^p \text{SOL}_p^p(C_f)}{\text{ALG}_p^{p-1}(C_1)} \\ &\le \frac{2}{3} \cdot 9^p \frac{\max_{C'} \left[\text{MRS}_p^p(C') - 2^p \text{SOL}_p^p(C' \setminus C_f)\right] - 18^p \text{SOL}_p^p(C_f)}{\text{ALG}_p^{p-1}(C_1)} \\ &\le \frac{2}{3} \cdot 9^p \frac{\max_{C'} \left[\text{MRS}_p^p(C') - 2^p \text{SOL}_p^p(C')\right]}{\text{ALG}_p^{p-1}(C_1)}. \end{aligned}$$

Using the same analysis as in Lemma 3.9 we can upper bound this final quantity by $18p^{1/p} \cdot \text{MR}$, proving the lemma. □

Theorem 6.1 follows from Lemmas 6.3 and 6.5.

7 UNIVERSAL k-CENTER WITH FIXED CLIENTS

In this section, we discuss how to extend the proof of Theorem 4.1 to prove the following theorem:

Theorem 7.1. *There exists a $(9, 3)$-approximate algorithm for universal k-center with fixed clients.*

Proof. To extend the proof of Theorem 4.1 to the case where fixed clients are present, let $\text{APX}(C')$ denote the cost of a 3-approximation to the k-center problem with client set C'; it is well known how to compute $\text{APX}(C')$ in polynomial time [33]. A solution with regret r must be within distance $r_j := \text{APX}(C_s \cup \{j\}) + r$ of client j, otherwise in realization $C_s \cup \{j\}$ the solution has regret larger than r due to client j. The same algorithm as in the proof of Theorem 4.1 using this definition of r_j finds ALG within distance $3r_j = 3 \cdot \text{APX}(C_s \cup \{j\}) + 3 \cdot \text{MR} \le 9 \cdot \text{OPT}(C_s \cup \{j\}) + 3\text{MR}$ of client j. $\text{OPT}(C') \ge \text{OPT}(C_s \cup \{j\})$ for any realization C' and any client $j \in C'$, so this solution is a $(9, 3)$-approximation. □

8 HARDNESS OF UNIVERSAL CLUSTERING FOR GENERAL METRICS

In this section, we give some hardness results to help contextualize the algorithmic results. Much like the hardness results for k-median, all our reductions are based on the NP-hardness of approximating set cover (or equivalently, dominating set) due to the natural relation between the two types of problems. We state our hardness results in terms of ℓ_p-clustering. Setting $p = 1$ gives hardness results for k-median, and setting $p = \infty$ (and using the convention $1/\infty = 0$ in the proofs as needed) gives hardness results for k-center.

8.1 Hardness of Approximating α

Theorem 8.1. *For all $p \ge 1$, finding an (α, β)-approximate solution for universal ℓ_p-clustering where $\alpha < 3$ is NP-hard.*

Proof. We will show that given a deterministic (α, β)-approximate algorithm where $\alpha < 3$, we can design an algorithm (using the (α, β)-approximate algorithm as a subroutine) that solves the set cover problem (i.e., finds a set cover of size k if one exists) giving the lemma by NP-hardness of set cover. The algorithm is as follows: Given an instance of set cover, construct the following instance of universal ℓ_p-clustering:

- For each element, there is a corresponding client in the universal ℓ_p-clustering instance.
- For each set S, there is a cluster center that is distance 1 from the clients corresponding to elements in S and 3 from other all clients.

Then, we just run the universal ℓ_p-clustering algorithm on this instance, and output the sets corresponding to cluster centers this algorithm buys.

Assume for the rest of the proof that a set cover of size k exists. Then the corresponding k cluster centers are as close as possible to every client, and are always an optimal solution. This gives that $\text{MR} = 0$ for this universal ℓ_p-clustering instance.

Now, suppose by contradiction that this algorithm does not solve the set cover problem. That is, for some set cover instance we run an (α, β)-approximate algorithm where $\alpha < 3$ on the produced ℓ_p-clustering instance, and it produces a solution ALG that does not choose cluster centers corresponding to a set cover. This means it is distance 3 from some client j. For realization $C' = \{j\}$, we

have by the definition of (α, β)-approximation:

$$\textsc{alg}(C') \leq \alpha \cdot \textsc{opt}(C') + \beta \cdot \textsc{mr} \implies 3 \leq \alpha \cdot 1 + \beta \cdot 0 = \alpha,$$

which is a contradiction, giving the lemma. □

Note that for, e.g., k-median, we can classically get an approximation ratio of less than 3. So this theorem shows that the universal version of the problem is harder, even if we are willing to use arbitrary large β.

8.2 Hardness of Approximating β

We give the following result on the hardness of universal ℓ_p-clustering.

Theorem 8.2. *For all $p \geq 1$, finding an (α, β)-approximate solution for universal ℓ_p-clustering where $\beta < 2$ is NP-hard.*

Proof. We will show that given a deterministic (α, β)-approximate algorithm where $\beta < 2$, we can design an algorithm (using the (α, β)-approximate algorithm as a subroutine) that solves the dominating set problem (i.e., outputs at most k vertices that are a dominating set of size k if a dominating set of size k exists) giving the lemma by NP-hardness of dominating set. The algorithm is as follows: Given an instance of dominating set $G = (V, E)$, construct the following instance of universal ℓ_p-clustering:

- For each vertex $v \in V$, there is a corresponding k-clique of clients in the universal ℓ_p-clustering instance.
- For each $(u, v) \in E$, connect all clients in u's corresponding clique to all those in v's.
- Impose the shortest path metric on the clients, where all edges are length 1.

Then, we just run the universal ℓ_p-clustering algorithm on this instance, and output the set of vertices corresponding to cluster centers this algorithm buys.

Assume for the rest of the proof that a dominating set of size k exists in the dominating set instance. Then, a dominating set of size k also exists in the constructed universal ℓ_p-clustering instance (where the cliques this set resides in correspond to the vertices in the dominating set in the original instance). Thus, there is a solution to the universal ℓ_p-clustering instance that covers all clients at distance at most 1.

We will first show this dominating set solution is a minimum regret solution. Given a dominating set solution, note that in any realization of the demands, $\textsc{opt}$ can cover k locations at distance 0, and must cover the rest of the clients at distance at least 1. Thus, to maximize the regret of a dominating set solution, we pick any k clients covered at distance 1 by the dominating set, and choose the realization including only these clients.

Now, consider any solution that is not a dominating set. For such a solution, there is some k-clique covered at distance 2. We can make such a solution incur regret $2k^{1/p}$ by including all k clients in this clique, with the optimal solution being to buy all cluster centers in this clique at cost 0. Thus, the dominating set solution is a minimum regret solution, and $\textsc{mr} = k^{1/p}$.

Now consider any (α, β)-approximation algorithm and suppose this algorithm when run on the reduced dominating set instance does not produce a dominating set solution while one exists. Consider the realization C' including only the clients in some k-clique covered at distance 2. By definition of (α, β)-approximation we get

$$\begin{aligned} \textsc{alg}(C') &\leq \alpha \cdot \textsc{opt}(C') + \beta \cdot \textsc{mr} \\ 2k^{1/p} &\leq 0 + \beta k^{1/p} \\ 2 &\leq \beta. \end{aligned} \tag{17}$$

If $\beta < 2$, then this is a contradiction, i.e., the algorithm will always output a dominating set of size k if one exists. Thus an (α, β)-approximation algorithm where $\beta < 2$ can be used to solve the dominating set problem, proving the theorem. □

9 HARDNESS OF UNIVERSAL CLUSTERING FOR EUCLIDEAN METRICS

9.1 Hardness of Approximating α

We can consider the special case of ℓ_p-clustering where the cluster center and client locations are all points in $\mathbb{R}^d$, and the metric is a ℓ_q-norm in $\mathbb{R}^d$. One might hope that, e.g., for $d = 2$, $\alpha = 1 + \epsilon$ is achievable since for the classic Euclidean k-median problem, a PTAS exists [5]. We show that there is still a lower bound on α even for ℓ_p-clustering in $\mathbb{R}^2$.

Theorem 9.1. *For all $p \geq 1$, finding an (α, β)-approximate solution for universal ℓ_p-clustering in $\mathbb{R}^2$ using the ℓ_q-norm where $\alpha < \frac{1+\sqrt{7}}{2}$ for $q = 2$ or $\alpha < 2$ for $q = 1, \infty$ is NP-hard.*

Proof. The hardness is via reduction from the discrete k-center problem in $\mathbb{R}^2$. Section 3 of Reference [46] shows how to reduce an instance of planar 3-SAT (which is NP-hard) to an instance of Euclidean k-center in $\mathbb{R}^2$ using the ℓ_q norm as the metric such that:

- For every client, the distance to the nearest cluster center is 1.
- There exists a k-cluster center solution that is distance 1 from all clients if the planar 3-SAT instance is satisfiable, and none exists if the instance is unsatisfiable.
- Any solution that is strictly less than distance $\alpha - \epsilon$ away from all clients can be converted in polynomial time to a solution within distance 1 of all clients for $\alpha = \frac{1+\sqrt{7}}{2}$ if $q = 2$, $\alpha = 2$ if $q = 1, \infty$.

We note that Reference [46]'s reduction is actually to the "continuous" version of the problem where every point in $\mathbb{R}^2$ can be chosen as a cluster center, including the points clients are located at. That is, if we use this reduction without modification, then the first property is not actually true (since the minimum distance is 0). However, in the proof of correctness for this construction [46] shows that (both for the optimal solution and any algorithmic solution) it suffices to only consider cluster centers located at the centers of a set of ℓ_p discs of radius 1 chosen such that every client lies on at least one of these discs and no client is contained within any of these discs. So, taking this reduction and then restricting the choice of cluster centers to the centers of these discs, we retrieve an instance with the desired properties.

Now, consider the corresponding instance as a universal ℓ_p-clustering instance. Like in the proof of Theorem 8.1, if the planar 3-SAT instance reduced from is satisfiable, then there exists a clustering solution that is as close as possible to every client, i.e., has regret 0. So MR = 0. Thus, an (α, β)-approximate clustering solution is within distance α of every client (in the realization where only client j appears, OPT is 1 so an α-approximate solution must be within distance α of this client). In turn, using the properties of the reduced clustering instance, an (α, β)-approximation where α is less than the lower bound given in Reference [46] can be converted into an algorithm that solves planar 3-SAT. □

9.2 Hardness of Approximating β

We can also show that $\beta = 1$ is NP-hard in $\mathbb{R}^2$ using a similar reduction:

Theorem 9.2. *For all $p \geq 1$, finding an (α, β)-approximate solution for universal ℓ_p-clustering in $\mathbb{R}^2$ using the ℓ_q-norm where $\beta = 1$ for $q = 1, 2, \infty$ is NP-hard.*

Proof. We again use a reduction from planar 3-SAT due to Reference [46]. This time, we use the reductions in Section 4 of Reference [46] for simplicity, which has the properties that:

- Every client is distance 0 from a co-located cluster center, and the distance to the second-closest cluster center is 1.
- There exists a k-cluster center solution that is distance 1 from all but k clients and distance 0 from k clients (the ones at the cluster centers) if the planar 3-SAT instance is satisfiable, and none exists if the instance is unsatisfiable.

Consider any instance reduced from a satisfiable planar 3-SAT instance. The solution in the resulting instance SOL^* with the second property above has regret $k^{1/p}$ (and in fact, this is MR): by the first property above, no solution can be less than distance 1 away from any clients other than the k clients co-located with its cluster centers. In turn, the regret of the SOL^* against any adversarial SOL is maximized by the realization C' only including the clients co-located with the k cluster centers in the SOL. We then get $\text{SOL}^*(C') - \text{SOL}(C') = k^{1/p} - 0 = k^{1/p}$.

Now consider an arbitrary $(\alpha, 1)$-approximate universal solution ALG in this instance. Consider any set of k clients C' not co-located with ALG's cluster centers. $\text{OPT}(C') = 0$, so we get $\text{ALG}(C') \leq \alpha \cdot \text{OPT}(C') + \text{MR} = \text{MR} \leq k^{1/p}$. ALG is distance at least 1 from all clients in C' by construction, so this only holds if ALG is distance 1 from all clients in C'. This gives that ALG is distance 1 from all but k clients (those co-located with cluster centers in ALG), and distance 0 from the remaining clients. In turn, ALG satisfies the property of a solution corresponding to a satisfying assignment to the planar 3-SAT instance. This shows that an $(\alpha, 1)$-approximate universal solution to ℓ_p-clustering in $\mathbb{R}^2$ can be used to solve planar 3-SAT. □

10 FUTURE DIRECTIONS

In this article, we gave the first universal algorithms for clustering problems: k-median, k-means, and k-center (and their generalization to ℓ_p-clustering). While we achieve constant approximation guarantees for these problems, the actual constants are orders of magnitude larger than the best (non-universal) approximations known for these problems. In part to ensure clarity of presentation, we did not attempt to optimize these constants. But it is unlikely that our techniques will lead to *small* constants for the k-median and k-means problems (although, interestingly, we got small constants for k-center). However, we show that in general it is **NP**-hard to find an (α, β)-approximation algorithm for a universal clustering problem where α matches the approximation factor for the standard clustering problem. Therefore, it is not entirely clear what one should expect: *are there universal algorithms for clustering with approximation factors of the same order as the classical (non-universal) bounds?*

One possible approach to improving the constants is considering algorithms that use more than k cluster centers. For example, our $(9^p, \frac{2}{3} \cdot 9^p)$-approximation for ℓ_p^p-clustering with discounts can easily be improved to an $(3^p, 3^p)$-approximation if it is allowed to use $2k - 1$ cluster centers. This immediately improves all constants in the article. For example, our $(27, 49)$-approximation for universal k-median becomes a $(9, 18)$-approximation if it is allowed to use $2k - 1$ cluster centers. Unfortunately, our lower bounds on α, β apply even if the algorithm is allowed to use $(1 - \epsilon)k \ln n$ cluster centers, but it is an interesting problem to show that, e.g., using $(1 + \epsilon)k \ln n$ cluster centers allows one to beat either bound.

Another open research direction pertains to Euclidean clustering. Here, we showed that in $\mathbb{R}^d$ for $d \geq 2$, α needs to be bounded away from 1, which is in stark contrast to non-universal clustering problems that admit PTASes in constant-dimension Euclidean space. But, for $d = 1$, i.e., for universal clustering on a line, the picture is not as clear. On a line, the lower bounds on α are no longer valid, which brings forth the possibility of (non-bicriteria) approximations of regret. Indeed,

it is known that there is 2-approximation for universal k-median on a line [38], and even better, an *optimal* algorithm for universal k-center on a line [7]. This raises the natural question: *can we design a PTAS for the universal k-median problem on a line?*

APPENDICES

A RELATIONS BETWEEN VECTOR NORMS

For completeness, we give a proof of the following well-known fact that relates vector ℓ_p norms.

Fact A.1. *For any $1 \le p \le q$ and $x \in \mathbb{R}^n$, we have*

$$||x||_q \le ||x||_p \le n^{1/p-1/q} ||x||_q.$$

Proof. For all p, repeatedly applying Minkowski's inequality we have the following:

$$||x||_p = \left(\sum_{i=1}^{n} |x|^p\right)^{1/p} \le \left(\sum_{i=1}^{n-1} |x|^p\right)^{1/p} + |x_n| \le \left(\sum_{i=1}^{n-2} |x|^p\right)^{1/p} + |x_{n-1}| + |x_n| \le \cdots \le \sum_{i=1}^{n} |x_i| = ||x||_1.$$

Then we bound $||x||_q$ by $||x||_p$ as follows:

$$||x||_q = \left(\sum_{i=1}^{n} |x|^q\right)^{1/q} = \left(\left(\sum_{i=1}^{n} (|x|^p)^{q/p}\right)^{p/q}\right)^{1/p} \le \left(\sum_{i=1}^{n} |x|^p\right)^{1/p} = ||x||_p.$$

The inequality is by applying $||x'||_{q/p} \le ||x'||_1$ to the vector x' with entries $x'_i = |x_i|^p$. To bound $||x||_p$ by $||x||_q$, we invoke Holder's inequality as follows:

$$||x||_p^p = \sum_{i=1}^{n} |x|^p = \sum_{i=1}^{n} |x|^p \cdot 1 \le \left(\sum_{i=1}^{n} (|x_i|^p)^{q/p}\right)^{p/q} \left(\sum_{i=1}^{n} 1^{q/(q-p)}\right)^{1-p/q} = ||x||_q^p \cdot n^{1-p/q}.$$

Taking the pth root of this inequality gives the desired bound. □

The limiting behavior as $q \to \infty$ shows that $||x||_\infty \le ||x||_{c \log n} \le n^{1/c \log n} ||x||_\infty = 2^{1/c} ||x||_\infty$, i.e., that the ℓ_∞-norm and ℓ_p-norm for $p = \Omega(\log n)$ are within a constant factor.

B APPROXIMATIONS FOR ALL-CLIENTS INSTANCES ARE NOT UNIVERSAL

In this section, we demonstrate that even $(1+\epsilon)$-approximate (integer) solutions for the "all clients" instance for clustering problems are not guaranteed to be (α, β)-approximations for any finite α, β. This is in sharp contrast to the optimal (integer) solution, which is known to be a $(1, 2)$-approximation for a broad range of problems including the clustering problems considered in this article [38].

Consider an instance of universal 1-median with clients c_1, c_2 and cluster centers f_1, f_2. Both the cluster centers are at distance 1 from c_1, and at distances 0 and ϵ respectively from c_2 (see Figure 6). f_2 is a $(1+\epsilon)$-approximate solution for the realization containing both clients. In this instance, MR is 0 and so f_2 is not an (α, β)-approximation for any finite α, β due to the realization containing only c_2. The same example can be used for the ℓ_p-clustering objective for all $p \ge 1$ since f_2 has approximation factor $(1 + \epsilon^p)^{1/p} \le (1 + \epsilon)$ when all clients are present. In the case of k-center, f_2 is an optimal solution when all clients are present.

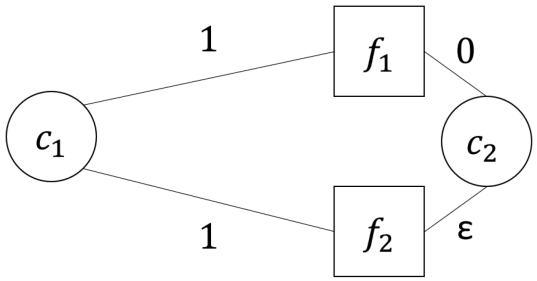

Fig. 6. Example where a $(1+\epsilon)$-approximation for all clients has no (α, β)-approximation guarantee, for any ℓ_p-clustering objective including k-center.

C ALGORITHMS FOR k-MEDIAN AND ℓ_p^p-CLUSTERING WITH DISCOUNTS

In this section, we prove Lemma 3.4, which states that there exists a $(9^p, \frac{2}{3} \cdot 9^p)$-approximation algorithm for the ℓ_p^p-clustering with discounts problem. As a corollary, by setting $p = 1$, we will obtain Lemma 2.3, which states that there exists a $(9, 6)$-approximation algorithm for the k-median with discounts problem. To prove Lemma 3.4, we will first use a technique due to Jain and Vazirani [34] to design a Lagrangian-preserving approximation for the ℓ_p^p-facility location with discounts problem. *ℓ_p^p-facility location with discounts (FLD)* is the same as ℓ_p^p-clustering with discounts, except rather than being restricted to buying k cluster centers, each cluster center has a cost f_i associated with buying it (discounts and cluster centers costs are not connected in any way).

C.1 Algorithm for ℓ_p^p-Facility Location with Discounts

Since FLD is a special case of non-metric facility location, we can consider the standard linear programming primal-dual formulation for the latter. The primal program is as follows:

$$\begin{aligned} & \min && \sum_{i \in F} f_i x_i + \sum_{i \in F, j \in C} (c_{ij}^p - r_j^p)^+ y_{ij} \\ \text{s.t.} \quad & \forall j \in C: && \sum_{i \in F} y_{ij} \geq 1 \\ & \forall i \in F, j \in C: && y_{ij} \leq x_i \\ & \forall i \in F: && x_i \geq 0 \\ & \forall i \in F, j \in C: && y_{ij} \geq 0. \end{aligned}$$

The dual program is as follows:

$$\begin{aligned} & \max && \sum_{j \in C} a_j \\ \text{s.t.} \quad & \forall i \in F, j \in C: && a_j - (c_{ij}^p - r_j^p)^+ \leq b_{ij} \\ & \forall i \in F: && \sum_{j \in C} b_{ij} \leq f_i \\ & \forall j \in C: && a_j \geq 0 \\ & \forall i \in F, j \in C: && b_{ij} \geq 0. \end{aligned}$$

We design a primal-dual algorithm for the FLD problem. This FLD algorithm operates in two phases. In both programs, all variables start out as 0.

In the first phase, we generate a dual solution. For each client j define a "time" variable t_j, which is initialized to 0. We grow the dual variables as follows: we increase the t_j uniformly. We grow the a_j such that for any j, at all times $a_j = (t_j - r_j^p)^+$ (or equivalently, all a_j start at 0, and we

increase all a_j at a uniform rate, but we only start growing a_j at time r_j^p). Each b_{ij} is set to the minimum feasible value, i.e., $(a_j - (c_{ij}^p - r_j^p)^+)^+$. If the constraint $\sum_{j \in C} b_{ij} \leq f_i$ is tight, then we stop increasing all t_j, a_j for which $b_{ij} = a_j - (c_{ij}^p - r_j^p)^+$, i.e., for the clients j that contributed to increasing the value $\sum_{j \in C} b_{ij}$ (we say these clients put weight on this cluster center). We continue this process until all t_j stop growing. Note that at any time the dual solution grown is always feasible.

In the second phase, consider a graph induced on the cluster centers whose constraints are tight, where we place an edge between cluster centers i, i' if there exists some client j that put weight on both cluster centers. Find a maximal independent set S of this graph and output this set of cluster centers. Let π be a map from clients to cluster centers such that $\pi(j)$ is the cluster center that made t_j stop increasing in the first phase of the algorithm. If $\pi(j) \in S$, then connect j to cluster center $\pi(j)$; otherwise, connect j to one of $\pi(j)$'s neighbors in the graph arbitrarily.

We can equivalently think of the algorithm as generating an integral primal solution where $x_i = 1$ for all $i \in S$ and $x_i = 0$ otherwise, and $y_{ij} = 1$ if j is connected to i and is $y_{ij} = 0$ otherwise. Based again on the technique of Reference [34], we can show the following Lemma holds:

Lemma C.1. *Let x, y be the primal solution and a, b be the dual solution generated by the above FLD algorithm. x, y satisfies*

$$\sum_{j \in C} \sum_{i \in F} (c_{ij}^p - 3^p r_j^p)^+ y_{ij} + 3^p \sum_{i \in F} f_i x_i \leq 3^p \sum_{j \in C} a_j.$$

Proof. Let $C^{(1)}$ be the set of clients in j such that $\pi(j) \in S$ and $C^{(2)} = C \setminus C^{(1)}$. For any $i \in S$, let C_i be the set of all clients j such that $\pi(j) = i$. Note that

$$\sum_{j \in C_i} a_j = \sum_{j \in C_i} [b_{ij} + (c_{ij} - r_j^p)^+] = \sum_{j \in C_i} (c_{ij} - r_j^p)^+ + f_i.$$

No client in $C^{(1)}$ contributes to the sum $\sum_{i \in F} b_{ij}$ for multiple i in S (because S is an independent set). This gives us

$$\begin{aligned}
\sum_{j \in C^{(1)}} \sum_{i \in F} (c_{ij}^p - 3^p r_j^p)^+ y_{ij} + \sum_{i \in F} f_i x_i &\leq \sum_{j \in C^{(1)}} \sum_{i \in F} (c_{ij}^p - r_j^p)^+ y_{ij} + \sum_{i \in F} f_i x_i \\
&= \sum_{i \in S} \left[f_i + \sum_{j \in C_i} (c_{ij}^p - r_j^p)^+ \right] \\
&= \sum_{i \in S} \sum_{j \in C_i} a_j \\
&= \sum_{j \in C^{(1)}} a_j. \qquad (18)
\end{aligned}$$

For each client j in $C^{(2)}$, j is connected to one of $\pi(j)$'s neighbors i. Since $\pi(j)$ and i are neighbors, there is some client j' that put weight on both $\pi(j)$ and i. Since j' put weight on $\pi(j)$ and thus $\pi(j)$ going tight would have stopped $t_{j'}$ from increasing, $t_{j'}$ stopped increasing before or when $\pi(j)$ went tight, which was when t_j stopped growing. Since all t_j start growing at the same time and grow uniformly, $t_{j'} \leq t_j$. Since j put weight on $\pi(j)$, we know $a_j - (c_{\pi(j)j} - r_j^p)^+ > 0$ and thus $(t_j - r_j^p)^+ - (c_{\pi(j)j} - r_j^p)^+ > 0$, implying $t_j \geq c_{\pi(j)j}^p$. Similarly, $t_{j'} \geq c_{ij'}^p, c_{\pi(j)j'}^p$. Triangle inequality

gives $c_{ij} \le c_{ij'} + c_{\pi(j)j'} + c_{\pi(j)j} \le 3t_j^{1/p}$. Then, we get

$$\sum_{j \in C^{(2)}} \sum_{i \in F} (c_{ij}^p - 3^p r_j^p)^+ y_{ij} \le \sum_{j \in C^{(2)}} (3^p t_j - 3^p r_j^p)^+ = 3^p \sum_{j \in C^{(2)}} (t_j - r_j^p)^+ = 3^p \sum_{j \in C^{(2)}} a_j. \tag{19}$$

Adding 3^p times Equation (18) to Equation (19) gives the Lemma. □

C.2 Algorithm for ℓ_p^p-Clustering with Discounts

We now move on to finding an algorithm for ℓ_p^p-clustering with discounts. We can represent the problem as a primal/dual linear program pair as follows. The primal program is as follows:

$$\begin{array}{lll} & \min & \displaystyle\sum_{i \in F, j \in C} (c_{ij}^p - r_j^p)^+ y_{ij} \\ \text{s.t.} & \forall j \in C: & \displaystyle\sum_{i \in F} y_{ij} \ge 1 \\ & \forall i \in F, j \in C: & y_{ij} \le x_i \\ & & \displaystyle\sum_{i \in F} x_i \le k \\ & \forall i \in F: & x_i \ge 0 \\ & \forall i \in F, j \in C: & y_{ij} \ge 0. \end{array}$$

The dual program is as follows:

$$\begin{array}{lll} & \max & \displaystyle\sum_{j \in C} a_j - kz \\ \text{s.t.} & \forall i \in F, j \in C: & a_j - (c_{ij}^p - r_j^p)^+ \le b_{ij} \\ & \forall i \in F: & \displaystyle\sum_{j \in C} b_{ij} \le z \\ & \forall j \in C: & a_j \ge 0 \\ & \forall i \in F, j \in C: & b_{ij} \ge 0. \end{array}$$

We now describe the algorithm we will use to prove Lemma 3.4, which uses the FLD algorithm from Section C.1 as a subroutine. By taking our ℓ_p-clustering with discounts instance and assigning all cluster centers the same cost z, we can produce a FLD instance. When $z = 0$, the FLD algorithm will either buy more than k cluster centers, or find a set of at most k cluster centers, in which case we can output that set. When $z = |C| \max_{i,j} c_{ij}$, the FLD algorithm will buy only one cluster center. Thus, for any ϵ such that $\log \frac{1}{\epsilon} = n^{O(1)}$, via bisection search using polynomially many runs of this algorithm we can find a value of z such that this algorithm buys a set of cluster centers S_1 of size $k_1 \ge k$ when cluster centers cost z and a set of cluster centers S_2 of size $k_2 \le k$ cluster centers when cluster centers cost $z + \epsilon$ (the bisection search starts with the range $[0, |C| \max_{i,j} c_{ij}]$ and in each iteration, determines how many cluster centers are bought when z is the midpoint value in its current range. It then recurses on the half $[a, b]$ of its current range, which maintains the invariant that when $z = a$, at least k cluster centers are bought and when $z = b$, at most k cluster centers are bought).

If either $k_1 = k$ or $k_2 = k$, then we output the corresponding cluster center set. Otherwise, we will randomly choose a solution that is roughly a combination of S_1 and S_2 (we will describe how to derandomize this process as is required to prove Lemma 2.3 later). Let ρ be the solution in $[0, 1]$ to $\rho k_1 + (1 - \rho)k_2 = k$, i.e., $\rho = \frac{k - k_2}{k_1 - k_2}$. Construct a set S_1' that consists of the closest cluster center in S_1 to each cluster center in S_2. If the size of S_1' is less than k_2, then add arbitrary cluster centers from $S_1 \setminus S_1'$ to S_1' until its size is k_2. Then, with probability ρ, let $S^* = S_1'$, otherwise let $S^* = S_2$. Then, sample a uniformly random subset of $k - k_2$ elements from $S_1 \setminus S_1'$ and add them to S^*. Then

output S^* (note that $S_1 \setminus S_1'$ is of size $k_1 - k_2$ so every element in $S_1 \setminus S_1'$ has probability ρ of being chosen).

Proof of Lemma 3.4. Note that if the FLD algorithm ever outputs a solution that buys exactly k cluster centers, then by Lemma C.1 we get that for the LP solution x, y encoding this solution and a dual solution a:

$$\sum_{j\in C}\sum_{i\in F}(c_{ij}^p - 3^p r_j^p)^+ y_{ij} + 3^p \sum_{i\in F} f_i x_i \le 3^p \sum_{j\in C} a_j$$
$$\sum_{j\in C}\sum_{i\in F}(c_{ij}^p - 3^p r_j^p)^+ y_{ij} + 3^p kz \le 3^p \sum_{j\in C} a_j$$
$$\sum_{j\in C}\sum_{i\in F}(c_{ij}^p - 3^p r_j^p)^+ y_{ij} + \le 3^p \left[\sum_{j\in C} a_j - kz\right],$$

which by duality means that this solution is also a $(3^p, 3^p)$-approximation for the ℓ_p-clustering with discounts instance.

If bisection search never finds a solution with exactly k cluster centers, but instead a pair of solutions S_1, S_2 where $|S_1| > k, |S_2| < k$, then the idea is that the algorithm constructs a "bi-point" fractional solution from these solutions (i.e., constructs a fractional solution that is just a convex combination of the two integral solutions) and then rounds it.

Consider the primal/dual solutions $x^{(1)}, y^{(1)}, a^{(1)}$ and $x^{(2)}, y^{(2)}, a^{(2)}$ corresponding to S_1, S_2. By Lemma C.1 we get the following:

$$\sum_{j\in C}\sum_{i\in F}(c_{ij}^p - 3^p r_j^p)^+ y_{ij}^{(1)} + 3^p k_1 z \le 3^p \sum_{j\in C} a_j^{(1)}$$
$$\sum_{j\in C}\sum_{i\in F}(c_{ij}^p - 3^p r_j^p)^+ y_{ij}^{(2)} + 3^p k_2 (z+\epsilon) \le 3^p \sum_{j\in C} a_j^{(2)}.$$

By combining the two inequalities and choosing ϵ appropriately we can get that

$$\sum_{j\in C}\sum_{i\in F}(c_{ij}^p - 3^p r_j^p)^+ (\rho y_{ij}^{(1)} + (1-\rho) y_{ij}^{(2)}) \le (3^p + \epsilon') \left[\sum_{j\in C}(\rho a_j^{(1)} + (1-\rho) a_j^{(2)}) - kz\right]$$

for an ϵ' we will fix later.

Note that $\rho(x^{(1)}, y^{(1)}, a^{(1)}) + (1-\rho)(x^{(2)}, y^{(2)}, a^{(2)})$ and z form a feasible (fractional) primal/dual solution pair for the ℓ_p-clustering with discounts problem, and by the above inequality the primal solution is a $(3^p, 3^p + \epsilon')$-approximation.

Then, we round the convex combination of the two solutions as described above. Let c_j be the connection cost of client j in the rounded solution, and $c_j^{(1)}, c_j^{(2)}$ the connection cost of client j in solutions S_1, S_2. Then since $(3^p + \epsilon')(2 \cdot 3^{p-1} - \epsilon') < \frac{2}{3} \cdot 9^p$ for $\epsilon \in [0,1]$ to prove the lemma it suffices to show that for each client j the expected contribution to the objective using discount $9r_j$ for client j is at most $2 \cdot 3^{p-1} - \epsilon'$ times the contribution of client j to the primal solution's objective using discount $3r_j$. That is,

$$\mathbb{E}\left[\left(c_j^p - 9^p r_j^p\right)^+\right] \le (2 \cdot 3^{p-1} - \epsilon')\left[\rho\left(c_j^{(1)p} - 3^p r_j^p\right)^+ + (1-\rho)\left(c_j^{(2)p} - 3^p r_j^p\right)^+\right].$$

Suppose client j's nearest cluster center in S_1 is in S_1'. Then with probability ρ, j is connected to that cluster center at connection cost $c_j^{(1)}$, and with probability $1-\rho$ it is connected to the nearest

cluster center in S_2 at connection cost $c_j^{(2)}$. Then

$$\mathbb{E}\left[\left(c_j^p - 9^p r_j^p\right)^+\right] \le \mathbb{E}\left[\left(c_j^p - 3^p r_j^p\right)^+\right] = \rho\left(c_j^{(1)p} - 3^p r_j^p\right)^+ + (1-\rho)\left(c_j^{(2)p} - 3r_j^p\right)^+.$$

Suppose client j's nearest cluster center in S_1 (call it i_1) is not in S_1'. Note that each cluster center in $S_1 \setminus S_1'$ has probability ρ of being opened. Thus with probability ρ, we can upper bound c_j by the distance from j to i_1. If this does not happen, then let i_2 be j's nearest cluster center in S_2 and i_1' be the cluster center nearest to i_2 in S_1. One of i_1', i_2 must be opened, so we can bound j's connection cost by its connection cost to whichever is opened. Then one of three cases occurs:

- With probability ρ, j's nearest cluster center in S_1 is opened. Then c_j is at most the distance from j to i_1, i.e., $c_j^{(1)}$.
- With probability $(1-\rho)\rho$, j's nearest cluster center in S_1 is not opened and S_1' is opened. c_j is at most the distance from j to i_1'. Since i_1' is the cluster center closest to i_2 in S_1, the distance from i_1' to i_2 is at most the distance from i_1 to i_2, which is at most $c_j^{(1)} + c_j^{(2)}$. Then by triangle inequality, the distance from j to i_1' is at most $c_j^{(1)} + 2c_j^{(2)}$. Using the AMGM inequality, we get $c_{i_1'j}^p \le 3^{p-1}(c_j^{(1)p} + 2c_j^{(2)p})$.
- With probability $(1-\rho)^2$, j's nearest cluster center in S_1 is not opened and S_2 is opened. c_j is at most the distance from j to i_2, i.e., $c_j^{(2)}$.

Then we get

$$\begin{aligned}
&\mathbb{E}\left[\left(c_j - 9^p r_j^p\right)^+\right] \\
&\le \rho\left(c_j^{(1)p} - 9^p r_j^p\right)^+ + (1-\rho)^2\left(c_j^{(2)p} - 9^p r_j^p\right)^+ + (1-\rho)\rho\left(3^{p-1}\left(c_j^{(1)} + 2c_j^{(2)}\right) - 9^p r_j^p\right)^+ \\
&= \rho\left(c_j^{(1)p} - 9^p r_j^p\right)^+ + (1-\rho)^2\left(c_j^{(2)p} - 9^p r_j^p\right)^+ + 3^{p-1}(1-\rho)\rho\left(c_j^{(1)} + 2c_j^{(2)} - 3\cdot 3^p r_j^p\right)^+ \\
&\le \rho\left(c_j^{(1)} - 3^p r_j^p\right)^+ + (1-\rho)^2\left(c_j^{(2)} - 3^p r_j^p\right)^+ + 3^{p-1}(1-\rho)\rho\left(c_j^{(1)} - 3^p r_j^p\right)^+ \\
&\quad + 2\cdot 3^{p-1}(1-\rho)\rho\left(c_j^{(2)} - 3^p r_j^p\right)^+ \\
&= (3^{p-1}(1-\rho)+1)\left[\rho\left(c_j^{(1)} - 3^p r_j^p\right)^+\right] + (2\cdot 3^{p-1}\rho + 1 - \rho)\left[(1-\rho)\left(c_j^{(2)} - 3^p r_j^p\right)^+\right] \\
&\le (2\cdot 3^{p-1} - \min\{\rho, 1-\rho\})\left[\rho\left(c_j^{(1)} - 3^p r_j^p\right)^+ + (1-\rho)\left(c_j^{(2)} - 3^p r_j^p\right)^+\right] \\
&\le (2\cdot 3^{p-1} - \epsilon')\left[\rho\left(c_j^{(1)} - 3^p r_j^p\right)^+ + (1-\rho)\left(c_j^{(2)} - 3^p r_j^p\right)^+\right].
\end{aligned}$$

Where the last step is given by choosing ϵ' to be at most $\frac{1}{|F|}$, since $p = \frac{k-k_2}{k_1-k_2}$ and $1 \le k_2 < k < k_1 \le |F|$ and thus ρ and $1-\rho$ are both at least $\frac{1}{|F|}$. This gives the lemma, except that the algorithm is randomized. However, the randomized rounding scheme can easily be derandomized: first, we choose S^* to be whichever of S_1', S_2 has a lower expected objective. Then, to choose the remaining $k - k_2$ cluster centers to add to S^*, we add cluster centers by one by one. When we have c cluster centers left to add to S^*, we add the cluster center i from $S_1 \setminus S_1'$ that minimizes the expected objective achieved by $S^* \cup \{i\}$ and $c-1$ random cluster centers from $(S_1 \setminus S_1') \setminus (S^* \cup \{i\})$. Each step of the derandomization cannot increase the expected objective, so the derandomized algorithm achieves the guarantee of Lemma 3.4. □

Again, we note that Lemma 2.3 is obtained as a corollary of Lemma 3.4, where we set $p = 1$.

REFERENCES

[1] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. 2017. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computing*. 61–72. https://doi.org/10.1109/FOCS.2017.15

[2] N. Alon and Y. Azar. 1992. On-line steiner trees in the Euclidean plane. In *Proceedings of the 8th Annual Symposium on Computational Geometry*. 337–343.

[3] Barbara Anthony, Vineet Goyal, Anupam Gupta, and Viswanath Nagarajan. 2010. A plant location guide for the unsure: Approximation algorithms for min-max location problems. *Math. Operat. Res.* 35, 1 (Februrary 2010), 79–101. https://doi.org/10.1287/moor.1090.0428

[4] Aaron Archer, Ranjithkumar Rajagopalan, and David B. Shmoys. 2003. Lagrangian relaxation for the k-median problem: New insights and continuity properties. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, Giuseppe Di Battista and Uri Zwick (Eds.). Springer, Berlin, 31–42. https://doi.org/10.1007/978-3-540-39658-1_6

[5] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. 1998. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*. ACM, New York, NY, 106–113. https://doi.org/10.1145/276698.276718

[6] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. 2001. Local search heuristic for k-median and facility location problems. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*. ACM, New York, NY, 21–29. https://doi.org/10.1145/380752.380755

[7] I. Averbakh and Oded Berman. 1997. Minimax regret p-center location on a network with demand uncertainty. *Locat. Sci.* 5, 4 (1997), 247–254. https://doi.org/10.1016/S0966-8349(98)00033-3

[8] Igor Averbakh and Oded Berman. 2000. Minmax regret median location on a network under uncertainty. *INFORMS J. Comput.* 12, 2 (2000), 104–110. https://doi.org/10.1287/ijoc.12.2.104.11897

[9] D. Bertsimas and M. Grigni. 1989. Worst-case examples for the spacefilling curve heuristic for the Euclidean traveling salesman problem. *Operat. Res. Lett.* 8, 5 (October 1989), 241–244.

[10] Anand Bhalgat, Deeparnab Chakrabarty, and Sanjeev Khanna. 2011. Optimal lower bounds for universal and differentially private Steiner trees and TSPs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim (Eds.). Springer, Berlin, 75–86.

[11] Sayan Bhattacharya, Parinya Chalermsook, Kurt Mehlhorn, and Adrian Neumann. 1994. New approximability results for the robust k-median problem. In *Proceedings of the 14th Scandanavian Workshop on Algorithm Theory*. 51–60.

[12] Costas Busch, Chinmoy Dutta, Jaikumar Radhakrishnan, Rajmohan Rajaraman, and Srinivasagopalan Srivathsan. 2012. Split and join: Strong partitions and universal Steiner trees for graphs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*. 81–90.

[13] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. 2013. Steiner tree approximation via iterative randomized rounding. *J. ACM* 60, 1 (2013), 6:1–6:33.

[14] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM J. Comput.* 40, 6 (December 2011), 1740–1766. https://doi.org/10.1137/080733991

[15] Deeparnab Chakrabarty and Chaitanya Swamy. 2019. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC'19)*. Association for Computing Machinery, New York, NY, 126–137. https://doi.org/10.1145/3313276.3316322

[16] Moses Charikar, Chandra Chekuri, and Martin Pál. 2005. Sampling bounds for stochastic optimization. In *Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th International Conference on Randamization and Computation: Algorithms and Techniques (APPROX'05/RANDOM'05)*. Springer-Verlag, Berlin, 257–269. https://doi.org/10.1007/11538462_22

[17] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. 1999. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC'99)*. ACM, New York, NY, 1–10. https://doi.org/10.1145/301250.301257

[18] Chandra Chekuri. 2007. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News* 38, 3 (2007), 106–129.

[19] Kedar Dhamdhere, Vineet Goyal, R. Ravi, and Mohit Singh. 2005. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. 367–378.

[20] Uriel Feige, Kamal Jain, Mohammad Mahdian, and Vahab S. Mirrokni. 2007. Robust combinatorial optimization with exponential scenarios. In *Proceedings of the Integer Programming and Combinatorial Optimization*. 439–453.

[21] Teofilo F. Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.* 38 (1985), 293–306.

[22] Igor Gorodezky, Robert D. Kleinberg, David B. Shmoys, and Gwen Spencer. 2010. Improved lower bounds for the universal and a priori TSP. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, Maria Serna, Ronen Shaltiel, Klaus Jansen, and José Rolim (Eds.). Springer, Berlin, 178–191.

[23] F. Grandoni, A. Gupta, S. Leonardi, P. Miettinen, P. Sankowski, and M. Singh. 2008. Set covering with our eyes closed. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*.

[24] Martin Grötschel, László Lovász, and Alexander Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1, 2 (1981), 169–197.

[25] Sudipto Guha and Kamesh Munagala. 2009. Exceeding expectations and clustering uncertain data. In *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'09)*. Association for Computing Machinery, New York, NY, 269–278. https://doi.org/10.1145/1559795.1559836

[26] Anupam Gupta, Mohammad T. Hajiaghayi, and Harald Räcke. 2006. Oblivious network design. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm (SODA'06)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 970–979. http://dl.acm.org/citation.cfm?id=1109557.1109665.

[27] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. 2014. Thresholded covering algorithms for robust and max-min optimization. *Math. Program.* 146, 1-2 (2014), 583–615.

[28] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. 2016. Robust and MaxMin optimization under matroid and knapsack uncertainty sets. *ACM Trans. Algor.* 12, 1 (2016), 10:1–10:21.

[29] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. 2004. Boosted sampling: Approximation algorithms for stochastic optimization. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*. ACM, New York, NY, 417–426. https://doi.org/10.1145/1007352.1007419

[30] Anupam Gupta and Kanat Tangwongsan. 2008. Simpler analyses of local search algorithms for facility location. arXiv:0809.2554. Retrieved from https://arxiv.org/abs/0809.2554.

[31] Mohammad T. Hajiaghayi, Robert Kleinberg, and Tom Leighton. 2006. Improved lower and upper bounds for universal TSP in planar metrics. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*. 649–658.

[32] Dorit S. Hochbaum and David B. Shmoys. 1985. A best possible heuristic for the k-center problem. *Math. Operat. Res.* 10, 2 (May 1985), 180–184. https://doi.org/10.1287/moor.10.2.180

[33] Dorit S. Hochbaum and David B. Shmoys. 1986. A unified approach to approximation algorithms for bottleneck problems. *J. ACM* 33, 3 (May 1986), 533–550. https://doi.org/10.1145/5925.5933

[34] Kamal Jain and Vijay V. Vazirani. 2001. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM* 48, 2 (March 2001), 274–296. https://doi.org/10.1145/375827.375845

[35] L. Jia, G. Lin, G. Noubir, R. Rajaraman, and R. Sundaram. 2005. Universal algorithms for TSP, Steiner tree, and set cover. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*.

[36] Lujun Jia, Guevara Noubir, Rajmohan Rajaraman, and Ravi Sundaram. 2006. GIST: Group-independent spanning tree for data aggregation in dense sensor networks. In *Distributed Computing in Sensor Systems*, Phillip B. Gibbons, Tarek Abdelzaher, James Aspnes, and Ramesh Rao (Eds.). Springer, Berlin, 282–304.

[37] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. 2002. A local search approximation algorithm for k-means clustering. In *Proceedings of the 18th Annual Symposium on Computational Geometry (SCG'02)*. ACM, New York, NY, 10–18. https://doi.org/10.1145/513400.513402

[38] Adam Kasperski and Pawel Zielinski. 2007. On the existence of an FPTAS for minmax regret combinatorial optimization problems with interval data. *Operat. Res. Lett.* 35, 4 (2007), 525–532.

[39] Rohit Khandekar, Guy Kortsarz, Vahab S. Mirrokni, and Mohammad R. Salavatipour. 2013. Two-stage robust network design with exponential scenarios. *Algorithmica* 65, 2 (2013), 391–408.

[40] Samir Khuller and Yoram J. Sussmann. 2000. The capacitated k-center problem. *SIAM J. Discr. Math.* 13, 3 (2000), 403–418. https://doi.org/10.1137/S0895480197329776

[41] Stavros G. Kolliopoulos and Satish Rao. 1999. A nearly linear-time approximation scheme for the Euclidean k-median problem. In *Proceedings of the Annual European Symposium on Algorithms (ESA'99)*, Jaroslav Nešetřil (Ed.). Springer, Berlin, 378–389.

[42] Panos Kouvelis and Gang Yu. 1997. *Robust 1-Median Location Problems: Dynamic Aspects and Uncertainty*. Springer US, Boston, MA, 193–240. https://doi.org/10.1007/978-1-4757-2620-6_6

[43] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. 2004. A simple linear time $(1 + \epsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*. 454–462.

[44] Shi Li and Ola Svensson. 2013. Approximating k-median via pseudo-approximation. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*. 901–910.

[45] Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28, 2 (1982), 129–136.

[46] Stuart G. Mentzer. 2016. Approximability of Metric Clustering Problems (unpublished).

[47] Viswanath Nagarajan, Baruch Schieber, and Hadas Shachnai. 2013. The Euclidean k-supplier problem. In *Integer Programming and Combinatorial Optimization*, Michel Goemans and José Correa (Eds.). Springer, Berlin, 290–301.
[48] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Math. Program.* 14, 1 (1978), 265–294. https://doi.org/10.1007/BF01588971
[49] Loren K. Platzman and John J. Bartholdi, III. 1989. Spacefilling curves and the planar travelling salesman problem. *J. ACM* 36, 4 (October 1989), 719–737. https://doi.org/10.1145/76359.76361
[50] Frans Schalekamp and David B. Shmoys. 2008. Algorithms for the universal and a priori TSP. *Operat. Res. Lett.* 36, 1 (2008), 1–3. https://doi.org/10.1016/j.orl.2007.04.009
[51] David B. Shmoys and Chaitanya Swamy. 2006. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM* 53, 6 (November 2006), 978–1012. https://doi.org/10.1145/1217856.1217860
[52] Chaitanya Swamy and David B. Shmoys. 2006. Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News* 37, 1 (2006), 33–46.
[53] Chaitanya Swamy and David B. Shmoys. 2012. Sampling-based approximation algorithms for multistage stochastic optimization. *SIAM J. Comput.* 41, 4 (2012), 975–1004.

Received 14 October 2021; revised 17 October 2022; accepted 6 November 2022