Universal Algorithms for Clustering Problems

Arun Ganesh

Department of Computer Science, UC Berkeley, USA

Bruce M. Maggs

Department of Computer Science, Duke University, USA

Emerald Innovations, USA

Debmalya Panigrahi

Department of Computer Science, Duke University, USA

Abstract

This paper 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 SOL for a clustering problem is said to be an (α, β)-approximation if for all subsets of clients C', it satisfies $\text{SOL}(C') \le \alpha \cdot \text{OPT}(C') + \beta \cdot \text{MR}$, where $\text{OPT}(C')$ is the cost of the optimal solution for clients C' and 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.

2012 ACM Subject Classification Theory of computation → Facility location and clustering

Keywords and phrases universal algorithms, clustering, k-median, k-means, k-center

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.59

Category Invited Talk

Related Version *Full Version*: https://arxiv.org/abs/2105.02363

Funding *Arun Ganesh*: Supported in part by NSF Award CCF-1535989.
Bruce M. Maggs: Supported in part by NSF grant CCF-1535972.
Debmalya Panigrahi: 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.

1 Introduction

In *universal*[1] approximation (e.g., [8, 9, 10, 16, 20, 22, 27, 39, 40]), the algorithm is presented with a set of *potential* input points and must produce a solution. After seeing the solution,

[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 paper.

© Arun Ganesh, Bruce M. Maggs, and Debmalya Panigrahi;
licensed under Creative Commons License CC-BY 4.0

48th International Colloquium on Automata, Languages, and Programming (ICALP 2021).
Editors: Nikhil Bansal, Emanuela Merelli, and James Worrell; Article No. 59; pp. 59:1–59:20
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 paper 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 $\textsc{sol}$ such that for all realizations (i.e., subsets of clients) C', $\textsc{sol}(C') \le \alpha \cdot \textsc{opt}(C')$, where $\textsc{sol}(C')$ denotes $\textsc{sol}$'s cost in realization C' and $\textsc{opt}(C')$ 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 $\textsc{sol}$ 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 $\textsc{sol}$. Then, $\textsc{sol}(C') > 0$ but $\textsc{opt}(C') = 0$. Since it is not possible to provide a strict approximation guarantee for every realization, we 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. The solution that minimizes regret is called the *minimum regret solution*, or $\textsc{mrs}$ for short, and its regret is termed *minimum regret* or $\textsc{mr}$. More formally, $\textsc{mr} = \min_{\textsc{sol}} \max_{C'} [\textsc{sol}(C') - \textsc{opt}(C')]$. We now seek a solution $\textsc{sol}$ that achieves, for all input realizations C', $\textsc{sol}(C') - \textsc{opt}(C') \le \textsc{mr}$, i.e., $\textsc{sol}(C') \le \textsc{opt}(C') + \textsc{mr}$. But, obtaining such a solution turns out to be **NP**-hard for many problems, and one has to settle for an approximation: $\textsc{sol}(C') \le \alpha \cdot \textsc{opt}(C') + \beta \cdot \textsc{mr}$. The algorithm 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 $\textsc{mr}$ (the distance between any two points) in some realization where $\textsc{opt}(C') = 0$ and only one client appears (in which case paying $\textsc{mr}$ 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 paper, the input is a metric space on all the potential client locations C and cluster centers F. The special case where $F = C$ has also been studied in the clustering literature, e.g., in [23, 14], 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 $F = C$, 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 $F = C$.

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 $\textsc{sol}$. 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., $\textsc{cost}(j, \textsc{sol}) = \min_{i \in \textsc{sol}} c_{ij}$. The clustering problems differ in how these costs are combined into the overall minimization objective. The respective objectives are given below:

- k**-median** (e.g., [14, 25, 5, 34, 11]): $\textsc{sol}(C') = \sum_{j \in C'} \textsc{cost}(j, \textsc{sol})$.
- k**-center** (e.g., [23, 15, 24, 30, 37]): $\textsc{sol}(C') = \max_{j \in C'} \textsc{cost}(j, \textsc{sol})$.
- k**-means** (e.g., [35, 28, 33, 21, 1]): $\textsc{sol}(C') = \sqrt{\sum_{j \in C'} \textsc{cost}(j, \textsc{sol})^2}$.

We also consider ℓ_p**-clustering** (e.g., [21]) 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., $\textsc{sol}(C') = \left(\sum_{j \in C'} \textsc{cost}(j, \textsc{sol})^p\right)^{1/p}$. 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 $p > \log n$, 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.

▶ **Theorem 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, 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:

▶ **Theorem 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 paper 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 [4, 31, 33, 37, 1].

► **Theorem 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 (resp., β) are independent of the value of β (resp., 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 [29]; 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 paper 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 [29] does not generalize to *any* regret guarantee for the optimal *fractional* solution. Furthermore, for all problems considered in this paper, even $(1+\epsilon)$-approximate (integer) solutions for the "all clients" instance are not guaranteed to be (α, β)-approximations for any finite α, β. These observations fundamentally distinguish universal approximations for **NP**-hard problems like the clustering problems in this paper from those in **P**, and require us to develop new techniques for universal approximations.

In this paper, 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) as well as the extension to universal clustering with fixed clients (Theorem 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 is given by

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

(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.)

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. It suffices to design a separation oracle for the new set of constraints $\mathbf{c}(I)\cdot\mathbf{x} \leq \text{OPT}(I) + \mathbf{r}$, $\forall I$. 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, as well as the setting with fixed clients, the situation is more complex, but can still be reduced to submodular maximization.

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. 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. 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.

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 [19], 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 for ℓ_p^p-clustering 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, extra work is needed to relate the ℓ_p and ℓ_p^p objectives. 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 these results, 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, starting with the seminal work of Jia *et al.* [26] (later improved by [20]), with following work giving better approximations for Euclidean metrics [39], minor-free metrics [22], and tree metrics [40]. The universal metric Steiner tree problem was also considered by Jia *et al.* [26], with nearly matching lower bounds [2, 26, 9]. The problem has also been considered for general graphs and minor-free graphs [10]. Finally, for universal (weighted) set cover, Jia *et al.* [26] (see also [17]) provide an algorithm and an almost matching lower bound.

The problem of minimizing regret has been studied in the context of robust optimization, with a focus on tree metrics. The robust 1-median problem was introduced for tree metrics by Kouvelis and Yu in [32] and several faster algorithms and for general metrics were developed in the following years (e.g. see [7]). For robust k-center, Averbakh and Berman[7] gave a reduction to ordinary k-center problems, which are tractable on tree metrics.

Roadmap. We present the constant approximation algorithms (Theorem 1) for universal k-median, a sketch for k-means, and k-center in Sections 2, 4, and 5 respectively. The k-means result is given in full detail as a more general ℓ_p-clustering result in the full paper. In describing these algorithms, we defer the clustering with discounts algorithms used in the rounding to the appendix. We also give the extension to universal clustering with fixed clients for k-median in Section 3, with the extensions for k-means and k-center in the full paper. Finally, the hardness results (Theorem 3) appear in Section 6.

2 Universal k-Median

In this section, we prove the following theorem:

▶ **Theorem 4.** *There exists a $(27, 49)$-approximate universal algorithm for the k-median problem.*

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 $f : 2^E \rightarrow \mathbb{R}_0^+$ 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 5** (Fisher *et al.* [38]). *For the problem of finding $S \subseteq E$ that maximizes a monotone submodular function $f : 2^E \rightarrow \mathbb{R}_0^+$, 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.

k-median with Discounts. The second component of our framework is a rounding algorithm that employs the k-median with discounts problem, which we define next. 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, [19] 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 6.** *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 the full paper. We note that when all r_j are equal, the constants in [19] can be improved (see e.g. [13]); 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., [25]) is given by:

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

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:

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

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.

In order 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 and submodular, and efficiently computable.

▶ **Lemma 7.** *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 $S = \emptyset$, 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:

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

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 [18]), 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
- Outputs 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 7 and Theorem 5. 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 $f_y(S) > r$, 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 8.** *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 8. We will use the following property of k-median, shown by Archer *et al.* [3].

▶ **Lemma 9** ([3]). *The integrality gap of the natural* LP *relaxation of the k-median problem is at most* 3.

Lemmas 8 and 9 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 approximates $\text{OPT}(C')$ by Eq. 2. This suggests minimizing the following objective instead: $\max_{C'}[\text{SOL}(C') - 3 \cdot \text{FRAC}(C')]$. Minimizing this objective is equivalent to the k-median with discounts problem, where the discount for client j is $3f_j$. This allows us to invoke Lemma 6 for the k-median with discounts problem.

Thus, our overall algorithm is as follows. First, use Lemma 8 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 6 on this instance to obtain the final solution to the universal k-median problem. Theorem 4 follows using the above lemmas; we defer the proof to the full paper.

3 Universal k-Median with Fixed Clients

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

▶ **Theorem 10.** *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 [34] for the k-median problem, and appropriate choice of both ϵ parameters, we obtain the following corollary from Theorem 10.

▶ **Corollary 11.** *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 follows similarly to the previous section. In Section 3.2, we show how to find a good fractional solution by approximately solving a linear program. In Section 3.3, we describe how to round the fractional solution in a manner that preserves its regret guarantee within constant factors. Similar techniques in conjunction with the techniques in Sections 4 and 5 are used for the universal k-means and k-center problems with fixed clients; due to space constraints, we only focus on universal k-median with fixed clients here.

3.1 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 [38] and more formally in [12].

▶ **Theorem 12.** *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 which 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 13.** *If there exists a* γ*-approximation algorithm for the* ℓ_p*-clustering problem, there exists a* γ*-approximation for the incremental* ℓ_p*-clustering problem.*

The lemma follows by an approximation-preserving reduction between the two problems, which simply adds many clients to the locations of cluster centers in S, forcing any low-cost solution to place cluster centers at these locations even in the standard ℓ_p-clustering problem.

3.2 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 $\text{OPT}(C')$ denote the cost of the optimal solution for C'. The same LP we used for universal k-median applies here, except we remove constraints on regret corresponding to realizations $C' \not\supseteq C_f$. Recall that to design an approximate separation oracle, it suffices to find a realization approximately maximizing the regret of the fractional solution.

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 $\text{OPT}(C') = \min_{S: S \subseteq F, |S| = k} S(C')$, exactly deciding the feasibility of the constraints on regret in the LP is equivalent to deciding if the following holds:

$$\forall S : S \subseteq F, |S| = k : \quad \max_{C' : C_f \subseteq C' \subseteq C} \left[\sum_{j \in C'} \sum_{i \in F} c_{ij} y_{ij} - S(C') \right] \leq r. \tag{3}$$

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 Eq. (3) as follows:

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

For fractional solution y, let

$$f_y(S) = \max_{C^*: C^* \subseteq C \setminus C_f} \left[\sum_{j \in C^*} \sum_{i \in F} c_{ij} y_{ij} - S(C^*) \right]. \quad (4)$$

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 submodular. But, the term $S(C_f)$ is not fixed with respect to S, so maximizing $f_y(S)$ does not suffice for separating the LP. 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):

$$\forall M \in \left\{0, 1, \ldots, |C_f| \max_{i,j} c_{ij}\right\} : \max_{S: S \subseteq F, |S|=k, S(C_f) \leq M} f_y(S) \leq M - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r. \quad (5)$$

Note that this set of inequalities is equivalent to Eq. (3), 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 $\epsilon > 0$, letting $Z_\epsilon := \left\{0, 1, 1+\epsilon, \ldots, (1+\epsilon)^{\lceil \log_{1+\epsilon}(|C_f| \max_{i,j} c_{ij}) \rceil + 1}\right\}$ we relax to the following polynomially large set of inequalities:

$$\forall M \in Z_\epsilon : \max_{S: S \subseteq F, |S|=k, S(C_f) \leq M} f_y(S) \leq M - \sum_{j \in C_f} \sum_{i \in F} c_{ij} y_{ij} + r. \quad (6)$$

Separating inequality Eq. (6) 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 Eq. (6) 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 3.1) 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 13. 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, due to the unpredictable behavior of A, $(F, \mathcal{I'}_M)$ 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 $S' \subseteq S$ 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 describe the modified greedy maximization algorithm GreedyMax that we use to try to separate one of the inequalities in Eq. (6), which uses a built-in membership oracle that ensures the above properties hold. Pseudocode is given in the full paper, 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 (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 the full paper, and we give an informal description of the algorithm here. SepOracle checks all constraints not involving the regret, 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 in Z_ϵ, 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. If 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.

Using the ellipsoid method where SepOracle is used as the separation oracle now gives the following lemma. The proof is deferred to the full paper.

► **Lemma 14.** *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.*

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

The rounding algorithm for universal k-median with fixed clients is almost identical to the rounding algorithm for universal k-median without fixed clients. The only difference is that in constructing a k-median with discounts problem, we give the fixed clients a discount of 0 rather than a discount of $3f_j$, as these clients will always appear and thus we want their connection cost to always factor into the cost of the k-median with discounts instance. The cost of a solution ALG to the k-median with discounts instance and the regret of ALG against an adversary with costs $3f_j$ now differs by $\sum_{j \in C_f} 3f_j$ (before, they were equal). However, as before $\sum_{j \in C_f} 3f_j$ is at most some constant times OPT(C_f) + MR, which lower bounds OPT(C') + MR for all realizations $C' \supseteq C_f$. So an analysis of the rounding similar to that in

Section 2 still allows us to prove Theorem 10, as the the offset $\sum_{j \in C_f} 3f_j$ (and multiples of it appearing in the analysis) can be absorbed into the (α, β)-approximation guarantee.

4 Universal k-means

In this section, we sketch our universal algorithm for k-means with the following guarantee:

▶ **Corollary 15.** *There exists a* $(108, 412)$*-approximate universal algorithm for the k-means problem.*

This follows as a special case of a more general ℓ_p-clustering result, given in the full paper; due to space constraints, we focus on k-means here.

Before describing further details of the universal k-means algorithm, we note a rather unusual feature of the universal clustering framework. Typically algorithms effectively optimize the ℓ_2^2 objective (i.e., sum of squared distances) instead of the k-means objective because these are equivalent in the following sense: an α-approximation for the k-means objective is equivalent to an α^2-approximation for the ℓ_2^2 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 outline our ℓ_p-clustering algorithm in the setting $p = 2$, which turns out to be much more challenging.

Similar to k-median, we use the primitive of an algorithm for the ℓ_p^p-clustering with discounts problem: 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[2] 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 full paper for details):

▶ **Lemma 16.** *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.*

The rest of this section is dedicated to sketching our algorithm for the universal k-means problem. As for k-median, we have two stages, the fractional algorithm and the rounding algorithm, that we sketch in the next two subsections.

[2] 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.

4.1 Universal k-means: Fractional Algorithm

Let us start by describing the fractional relaxation of the universal k-means problem[3] (again, P is the k-median polytope defined as in Section 2.1):

$$\min\{r : (x,y) \in P; \forall C' \subseteq C : \left(\sum_{j \in C'} \sum_i c_{ij}^2 y_{ij}\right)^{1/2} - \text{OPT}(C') \leq r\}, \tag{7}$$

As described earlier, when minimizing regret, the k-means and ℓ_2^2 objectives are no longer equivalent. For instance, recall that one of the key steps in Lemma 8 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_2^2 objective because of the linearity of $f_y(S)$ as a function of the realized clients once y and S are fixed. But, the k-means 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 $(2+\epsilon)^{1/2} - 1 < 1$, whereas with just the first client the regret is 1.

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. Let $\text{FRAC}_2, \text{FRAC}_2^2$ denote the k-means and ℓ_2^2-objectives of a given fractional solution, and S_2, S_2^2 the same for the solution using the set of cluster centers S. We can show that:

$$\max_{C' \subseteq C} \left[\text{FRAC}_2(C') - S_2(C')\right] \simeq_2 \max_Y \max_{C' \subseteq C : \text{FRAC}_2^2(C') \leq Y} \left[\frac{\text{FRAC}_2^2(C') - S_2^2(C')}{Y^{1/2}}\right],$$

where $\simeq_2$ denotes equality to within a factor of 2. In turn, by guessing the maximizing value of Y we can (approximately) reduce maximizing the difference in k-means objectives to maximizing the difference in ℓ_2^2 objectives, subject to the constraint $\text{FRAC}_2^2(C') \leq Y$.

A separation oracle then just needs to (approximately) compute $\max\{\text{FRAC}_2^2(C') - S_2^2(C') : C' \subseteq C, \text{FRAC}_2^2(C') \leq Y\}$ for each fixed (discretized) value of Y. To do so, we show that allowing an adversary to choose *fractional* realizations of clients does not give them an advantage.

▶ **Lemma 17.** *For any two solutions y, S, there exists a global maximum of $\text{FRAC}_2(\mathbf{I}) - S_2(\mathbf{I})$ over fractional realizations $\mathbf{I} \in [0,1]^C$ where all the clients are integral, i.e., $\mathbf{I} \in \{0,1\}^C$. Therefore,*

$$\max_{\mathbf{I} \in [0,1]^C} \left[\text{FRAC}_2(\mathbf{I}) - S_2(\mathbf{I})\right] = \max_{C' \subseteq C} \left[\text{FRAC}_2(C') - S_2(C')\right].$$

[3] The constraints are not simultaneously linear in y and r, although fixing r, we can write these constraints as $\sum_{j \in C'} \sum_i c_{ij}^p y_{ij} \leq (\text{OPT}(C') + r)^p$, 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.

We then show that $f_{y,Y}(S) := \max\{\text{FRAC}_2^2(\mathbf{I}) - S_2^2(\mathbf{I}) : \mathbf{I} \in [0,1]^C, \text{FRAC}_2^2(\mathbf{I}) \leq Y\}$ is a submodular function. Since we are allowed to use fractional clients, computing $f_{y,Y}(S)$ for a given S is a fractional knapsack problem which can be solved in polynomial time (whereas computing $\max\{\text{FRAC}_2^2(C') - S_2^2(C') : C' \subseteq C, \text{FRAC}_2^2(C') \leq Y\}$ requires solving an integer knapsack problem), giving an efficient separation oracle using the greedy algorithm for submodular maximization.

4.2 Universal k-Means: Rounding Algorithm

At a high level, we use the same strategy for rounding the fractional k-means solution as we did with k-median. Namely, we use Lemma 16 to 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 k-means objective, 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_2^2 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_2^2 objective to that in the k-means objective. We show that, roughly speaking, the realization that maximizes the regret of an algorithm ALG against a fixed solution SOL in both objectives is the same under a "farness" condition:

▶ **Lemma 18.** *Suppose* ALG *and* SOL *are two solutions to a k-means 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 2 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^* is a 1/2-maximizer of* $\text{ALG}_2(C') - \text{SOL}_2(C')$.

Given any solution SOL, it is easy to define a *virtual* solution $\widetilde{\text{SOL}}$ whose individual connection costs are bounded by 2 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 2 times SOL) in the ℓ_2^2 objective to its regret in the k-means objective.

5 Universal k-Center

In this section, we prove the following guarantee for universal k-center:

▶ **Theorem 19.** *There exists a $(3,3)$-approximate algorithm for the universal k-center problem.*

First, note that for every client j, its distance to the closest cluster center in the minimum regret solution MRS is at most $\text{MR}_j := \min_{i \in F} c_{ij} + \text{MR}$; otherwise, in the realization with only client j, MRS would have regret $>$ MR. We first 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. Since $\min_{i \in F} c_{ij}$ lower bounds $\text{OPT}(C')$ for any C' containing j, this gives a $(3,3)$-approximation. This algorithm actually satisfies a more general property: given *any* value r, it produces a set of cluster centers such that every client j is at a distance $\leq 3r_j$ from its closest cluster center, 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 20.** *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.

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 $r_{j_1} \leq r_{j_2}$. The distance from j_2 to i_1 is at most the length of the path (j_2, i', j_1, i_1) , which is at most $2r_{j_2} + r_{j_1} \leq 3r_{j_2}$. But, in this case j_2 would not have added a new cluster center i_2, thus arriving at a contradiction. ◀

Theorem 19 follows 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 can simply run the algorithm of Lemma 20 with r equal to each of these values, and then choose the solution corresponding to the smallest r that results in the algorithm using at most k cluster centers, which will be at most MR by Lemma 20.

We note that the greedy algorithm described above can be viewed as an extension of the k-center algorithm in [24] to a $(3,3)$-approximation for the "k-center with discounts" problem, where the discounts are the minimum distances $\min_{i\in F} c_{ij}$.

6 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.

The results in this section can be extended to Euclidean metrics by building off the reductions in [36], albeit with worse constants. Due to space constraints, we defer our results for Euclidean metrics to the full paper.

6.1 Hardness of Approximating α

▶ **Theorem 21.** *For all $p \geq 1$, finding an (α, β)-approximate solution for universal ℓ_p-clustering where $\alpha < 3$ is NP-hard.*

Proof. To prove the theorem, given an instance of set cover, we 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 which is distance 1 from the clients corresponding to elements in S and 3 from other all clients.

If there is a set cover of size k, the corresponding cluster centers are an optimal solution for any realization of clients, i.e. $\text{MR} = 0$. Furthermore, in any single-client realization, $\text{OPT} = 1$. So an (α, β)-approximate universal ℓ_p-clustering algorithm must find a solution within distance of α of every client to satisfy the regret guarantee in single-client realizations. If $\alpha < 3$, this implies it is distance 1 from every client, in which case its cluster centers also correspond to a set cover. So this algorithm can be used to find set covers of size at most k if they exist, which is an NP-hard task. ◀

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.

6.2 Hardness of Approximating β

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

▶ **Theorem 22.** *For all $p \geq 1$, finding an (α, β)-approximate solution for universal ℓ_p-clustering where $\beta < 2$ is NP-hard.*

Proof. Given an instance of dominating set $G = (V, E)$, we construct the following instance of universal ℓ_p-clustering:

- For each vertex $v \in V$, replace it with a k-clique.
- For each $(u, v) \in E$, add an edge from every vertex in u's clique to every vertex in v's clique.
- To turn this modified graph into a clustering instance, place a client and cluster center at each vertex, and impose the shortest path metric on the clients, where all edges are length 1.

Suppose a dominating set of size k exists in the modified graph (and thus in the original graph). Then the corresponding cluster centers are distance at most 1 from every client. Since any solution is distance 0 from at most k clients and distance at least 1 from all other clients, these cluster centers have regret at most $k^{1/p}$. It now suffices to prove the claim that any k cluster centers that aren't a dominating set have regret at least $2k^{1/p}$ in a realization where OPT has cost 0. The theorem follows since an (α, β)-approximate algorithm would produce a solution with cost at most $\beta k^{1/p}$ in any such realization, i.e. can be used to find dominating sets of size at most k if they exist when $\beta < 2$. The claim follows since if the cluster centers aren't a dominating set, there is a k-clique they are distance 2 or more from. The realization containing exactly the clients in this k-clique satisfies the desired properties. ◀

7 Future Directions

In this paper, we gave the first universal algorithms for clustering problems. 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. On the other hand, 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?*

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 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, there is 2-approximation for universal k-median on a line [29], and even better, an *optimal* algorithm for universal k-center on a line [6]. This raises the natural question: *can we design a PTAS for the universal k-median problem on a line?*

References

1. Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. 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*, pages 61–72, October 2017. doi:10.1109/FOCS.2017.15.
2. N. Alon and Y. Azar. On-line steiner trees in the Euclidean plane. In *Proceedings of the 8th Annual Symposium on Computational Geometry*, pages 337–343, 1992.
3. Aaron Archer, Ranjithkumar Rajagopalan, and David B. Shmoys. Lagrangian relaxation for the k-median problem: New insights and continuity properties. In Giuseppe Di Battista and Uri Zwick, editors, *Algorithms - ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings*, pages 31–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. URL: https://doi.org/10.1007/978-3-540-39658-1_6, doi:10.1007/978-3-540-39658-1_6.
4. Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 106–113, New York, NY, USA, 1998. ACM. URL: http://doi.acm.org/10.1145/276698.276718, doi:10.1145/276698.276718.
5. Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 21–29, New York, NY, USA, 2001. ACM. URL: http://doi.acm.org/10.1145/380752.380755, doi:10.1145/380752.380755.
6. I. Averbakh and Oded Berman. Minimax regret p-center location on a network with demand uncertainty. *Location Science*, 5(4):247 – 254, 1997. URL: http://www.sciencedirect.com/science/article/pii/S0966834998000333, doi:https://doi.org/10.1016/S0966-8349(98)00033-3.
7. Igor Averbakh and Oded Berman. Minmax regret median location on a network under uncertainty. *INFORMS Journal on Computing*, 12(2):104–110, 2000. URL: https://doi.org/10.1287/ijoc.12.2.104.11897, arXiv:https://doi.org/10.1287/ijoc.12.2.104.11897, doi:10.1287/ijoc.12.2.104.11897.
8. D. Bertsimas and M. Grigni. Worst-case examples for the spacefilling curve heuristic for the Euclidean traveling salesman problem. *Operations Research Letter*, 8(5):241–244, October 1989.
9. Anand Bhalgat, Deeparnab Chakrabarty, and Sanjeev Khanna. Optimal lower bounds for universal and differentially private Steiner trees and TSPs. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and*

Combinatorial Optimization. Algorithms and Techniques, pages 75–86, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

10 Costas Busch, Chinmoy Dutta, Jaikumar Radhakrishnan, Rajmohan Rajaraman, and Srinivasagopalan Srivathsan. Split and join: Strong partitions and universal Steiner trees for graphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 81–90, 2012.

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

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

13 Deeparnab Chakrabarty and Chaitanya Swamy. Approximation algorithms for minimum norm and ordered optimization problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 126–137, New York, NY, USA, 2019. Association for Computing Machinery. URL: `https://doi.org/10.1145/3313276.3316322`, `doi:10.1145/3313276.3316322`.

14 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 1–10, New York, NY, USA, 1999. ACM. URL: `http://doi.acm.org/10.1145/301250.301257`, `doi:10.1145/301250.301257`.

15 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

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

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

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

19 Sudipto Guha and Kamesh Munagala. Exceeding expectations and clustering uncertain data. In *Proceedings of the Twenty-Eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '09, page 269–278, New York, NY, USA, 2009. Association for Computing Machinery. URL: `https://doi.org/10.1145/1559795.1559836`, `doi:10.1145/1559795.1559836`.

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

21 Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *ArXiv*, abs/0809.2554, 2008.

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

23 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10(2):180–184, May 1985. URL: `http://dx.doi.org/10.1287/moor.10.2.180`, `doi:10.1287/moor.10.2.180`.

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

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

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

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

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

29 Adam Kasperski and Pawel Zielinski. On the existence of an FPTAS for minmax regret combinatorial optimization problems with interval data. *Oper. Res. Lett.*, 35:525–532, 2007.

30 Samir. Khuller and Yoram J. Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000. URL: https://doi.org/10.1137/S0895480197329776, arXiv:https://doi.org/10.1137/S0895480197329776, doi:10.1137/S0895480197329776.

31 Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem. In Jaroslav Nešetřil, editor, *Algorithms - ESA' 99*, pages 378–389, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

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

33 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. 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*, pages 454–462, 2004.

34 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, pages 901–910, 2013.

35 Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Information Theory*, 28:129–136, 1982.

36 Stuart G. Mentzer. Approximability of metric clustering problems. Unpublished manuscript, March 2016.

37 Viswanath Nagarajan, Baruch Schieber, and Hadas Shachnai. The Euclidean k-supplier problem. In Michel Goemans and José Correa, editors, *Integer Programming and Combinatorial Optimization*, pages 290–301, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

38 G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978. URL: https://doi.org/10.1007/BF01588971, doi:10.1007/BF01588971.

39 Loren K. Platzman and John J. Bartholdi, III. Spacefilling curves and the planar travelling salesman problem. *J. ACM*, 36(4):719–737, October 1989. URL: http://doi.acm.org/10.1145/76359.76361, doi:10.1145/76359.76361.

40 Frans Schalekamp and David B. Shmoys. Algorithms for the universal and a priori TSP. *Operations Research Letters*, 36(1):1–3, 2008. URL: http://www.sciencedirect.com/science/article/pii/S0167637707000697, doi:https://doi.org/10.1016/j.orl.2007.04.009.