A Constant-Factor Approximation Algorithm for Reconciliation k-Median

Joachim Spoerhase
University of Sheffield
UK

Kamyar Khodamoradi
University of British Columbia
BC, Canada

Benedikt Riegel
University of Stuttgart
Germany

Bruno Ordozgoiti
Queen Mary University of London
UK

Aristides Gionis
KTH Royal Institute of Technology
Sweden

Abstract

In the *reconciliation k-median* problem we ask to cluster a set of data points by picking k cluster centers so as to minimize the sum of distances of the data points to their cluster centers plus the sum of pairwise distances between the centers. The problem, which is a variant of classic k-median, aims to find a set of cluster centers that are not too far from each other, and it has applications, for example, when selecting a committee to deliberate on a controversial topic. This problem was introduced recently (Ordozgoiti and Gionis, 2019), and it was shown that a local-search-based algorithm is always within a factor $\mathcal{O}(k)$ of an optimum solution and performs well in practice.

In this paper, we demonstrate a close connection of reconciliation k-median to a variant of the *k-facility location problem*, in which each potential cluster center has an individual opening cost and we aim at minimizing the sum of client-center distances and the opening costs. This connection enables us to provide a new algorithm for reconciliation k-median that yields a *constant-factor approximation* (independent of k). We also provide a *sparsification* scheme that reduces the number of potential cluster centers to $\mathcal{O}(k)$ in order to substantially speed up approximation algorithms. We empirically compare our new algorithms with the previous local-search approach, showing improved performance and stability. In addition, we show how our sparsification approach helps to reduce computation time without significantly compromising the solution quality.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

1 INTRODUCTION

In many applications of data clustering, we are interested not only in partitioning the data into groups that contain similar points, but also to select *centers* that represent clusters in a meaningful way. The task of clustering a dataset while selecting cluster representatives can be achieved with formulations like *k-median* (Charikar et al., 2002; Jain and Vazirani, 2001) or *k-facility location* (Jain and Vazirani, 2001). Oftentimes, selecting the median of a cluster as the cluster center is not good enough for the application at hand, and it is desirable to consider additional criteria, such as, selecting cluster centers that have *low cost* or *high quality*, assuming that information to quantify these criteria is provided.

One such criterion for selecting cluster centers was recently introduced in the context of *reconciliation k-median* (Ordozgoiti and Gionis, 2019), where the goal is to find a clustering solution so that the data points are faithfully represented by the selected centers (or medians), while at the same time, the selected centers are *not too far apart* from each other. The reconciliation k-median problem is motivated by applications where it is desirable for the selected centers to form a *non-polarized* set of representatives. A concrete application scenario is exemplified by the task of selecting a set of k articles to summarize a large collection of news articles related to a controversial topic, and we want to minimize disagreement among the selected articles (say, for encouraging constructive deliberation), while ensuring a faithful representation of the news-articles collection.

More concretely, in the reconciliation k-median problem, we consider a set of clients C, a set of facilities F (not necessarily disjoint from C), a metric distance $d\colon (C \cup F)^2 \to \mathbb{R}_{\geq 0}$, a positive integer k, and a hyper-parameter $\lambda \geq 0$. The goal is to select a subset S of k facilities in F, so as to min-

imize the objective $\sum_{j\in C} d(j,S) + \lambda \sum_{i,i'\in S, i<i'} d(i,i')$, where $d(j,S) = \min_{i\in S} d(j,i)$. The first term models the cost of representing the clients in C by the set of selected facilities S.[1] We refer to this term as *representation cost* or *service cost*. The second term models the *disagreement cost* among the selected facilities in S. Notice that these two terms are conflicting in the optimization problem, and thus, the hyper-parameter λ is used to achieve a desirable trade-off.

In addition to introducing the reconciliation k-median problem and demonstrating interesting use cases, Ordozgoiti and Gionis (2019) analyzed the local-search algorithm proposed by Arya et al. (2004) for the k-median problem. It was shown that the local-search algorithm provides an $\mathcal{O}(k)$ approximation guarantee, if $C = F$, while it provides a constant-factor approximation if, in addition, the clusters in the obtained solution have size $\Theta(\lambda k)$. These results are not optimal, and improving the approximation guarantee was left open. Furthermore, while local search is a simple method to implement, it has quadratic complexity in each iteration, and thus, its practical applicability is limited.

Our Contributions. First, we obtain a constant-factor approximation for the reconciliation k-median problem in the general case, that is, our approximation factor is independent of k. Our analysis does not make any assumption on the sets of client and facilities, nor on the size of the output clusters. In particular, as we demonstrate in the Appendix, eliminating the condition $F = C$ may lead to the local search having unbounded approximation ratio.

Second, we show a close connection between reconciliation k-median and a variant of the previously studied *k-facility location problem* (Jain and Vazirani, 2001). Indeed, our algorithm relies on solving $|F|$ many instances of k-facility location and returning the solution that is best in terms of the reconciliation objective. Thus, the constant-factor approximation guarantee comes at the cost of increased run time complexity.

Third, we propose a sparsification scheme to address the increased complexity of the proposed method. Our sparsification method allows us to reduce the number of potential cluster centers to $\mathcal{O}(k)$, thus, providing significant speedup in scenarios where k is significantly smaller than $|F|$. For example, k is often considered to be a small constant. Our sparsification scheme is of independent value and it can be used in conjunction with other methods, for instance, to speed up the local-search algorithm.

Our Techniques. The main technical challenge in tackling reconciliation k-median (in comparison to classic k-median problems) lies in bounding the disagreement cost. The difficulty is that the contribution of a facility to the total disagreement is dependent of the choice of the other facilities. Our key idea is to identify an *anchor* facility i^* that can be thought of as a representative of the facilities in an (unknown) optimum solution. This allows us to estimate the contribution of any facility i to the total disagreement as being proportional to the distance $d(i,i^*)$ and hence making it independent of the choice of the other facilities. More specifically, we interpret this quantity as an individual opening cost and we aim at minimizing the sum of client-center distances and the opening costs thereby relating it to a variant of the k-facility location problem. Since the anchor i^* is not known upfront we have to create $|F|$ many candidate instances of k-facility location.

Our sparsification scheme relies on the idea of *adaptive sampling* (Arthur and Vassilvitskii, 2007) where clients are sampled sequentially with probability proportional to their distance to the previously sampled clients. Unfortunately, there are instances where vanilla adaptive sampling creates solutions of unbounded cost. This is due to the fact that the sampling process is biased towards distant clients thereby inherently tending to high disagreement cost. We observe that, interestingly enough, the ball of k facilities centered around the anchor i^* is good enough to serve the distant clients in configurations where these clients significantly contribute to the service cost. This insight allows us to create $|F|$ many sparsified candidate sets of size $O(k)$ by augmenting an adaptively sampled set of facilities with balls of size k. We show that one of these candidate sets contains a constant-factor approximate solution.

Our experimental evaluation shows the improved performance of the proposed methods, as well as the computational benefits of the sparsification approach.

Related Work. The k-median problem and its variants have been studied extensively both in theoretical computer science (Arya et al., 2004; Byrka et al., 2018; Charikar et al., 2002) and machine-learning literature (Ben-David, 2007). The k-median problem is $\mathcal{NP}$-hard (Garey and Johnson, 1979), and thus, many approximation algorithms and heuristics have been proposed. The first constant-factor approximation algorithm was obtained by Charikar et al. (2002), followed by a series of improvements relying on local search (Arya et al., 2004), LP rounding (Charikar and Li, 2012), or primal-dual methods (Byrka et al., 2018; Jain and Vazirani, 2001; Li and Svensson, 2016). Several variants of the problem have also been studied, for example, capacitated k-median (Demirci and Li, 2016), k-median with outliers (Chen, 2008), k-median with fairness constraints (Chierichetti et al., 2017), and connected facility location (Eisenbrand et al., 2008; Han et al., 2021). The problem we consider in this paper is similar in some ways to the connected facility location setting in the sense that in both, the objective is to minimize the connection cost of the clients plus the cost of certain structure on the selected facilities (a Steiner tree and a clique in the case of connected facility location and reconciliation k-median, respectively.)

[1] We interchangeably refer to cluster representatives as centers, medians, or facilities.

The reconciliation k-median problem, which is the focus of this paper, was proposed recently by Ordozgoiti and Gionis (2019), motivated by scenarios for reducing polarization for cluster representatives. They proposed a local-search algorithm, inspired by the work of Arya et al. (2004), which gives a $\mathcal{O}(k)$ approximation guarantee, in a special case, and constant-factor approximation under a mild condition on the cluster sizes. In this paper, we significantly improve on the result of Ordozgoiti and Gionis (2019), by providing a constant-factor approximation in the general case.

The technique of adaptive sampling has been introduced by Arthur and Vassilvitskii (2007) as an initialization procedure for k-means clustering, which has a provable quality of solution in expectation but is still remarkably simple and fast. Since then the method has proved useful in many aspects of clustering such as bi-criteria algorithms (Aggarwal et al., 2009), streaming algorithms (Ailon et al., 2009; Feldman et al., 2007), and core sets (Feldman et al., 2007; Langberg and Schulman, 2010).

2 CONSTANT-FACTOR APPROXIMATION

In this section we present the constant-factor approximation algorithm for the reconciliation k-median problem. The algorithm relies on a reduction to the *exact k-facility location* problem, a variant of *k-facility location* (Charikar and Li, 2012; Jain and Vazirani, 2001; Zhang, 2007).

Preliminaries and Problem Definitions

Definition 2.1. An instance of the RECONCILIATION k-MEDIAN problem is specified by a tuple (C, F, k, d, λ), where C and F are (possibly overlapping) finite sets, k is a positive integer, $d\colon (C \cup F)^2 \to \mathbb{R}_{\geq 0}$ is a metric, and $\lambda \geq 0$ is a parameter. The goal is to find a k-subset S of F minimizing the objective value

$$\text{cost}_{\text{RC}}(S) \coloneqq \sum_{j \in C} d(j, S) + \frac{\lambda}{2} \cdot \sum_{i, i' \in S} d(i, i')\,.$$

We denote by O_{RC} an optimal k-subset of facilities and its objective value by OPT_{RC}.

Definition 2.2. An instance of EXACT k-FACILITY LOCATION is specified by a tuple (C, F, k, d, f), where C, F are finite sets, k is a positive integer, $d\colon (C \cup F)^2 \to \mathbb{R}_{\geq 0}$ is a metric, and f_i is an opening cost for every facility $i \in F$. The goal is to find a k-subset S of F minimizing the objective value $\text{cost}_{k\text{-FL}}(S) \coloneqq \sum_{j \in C} d(j, S) + \sum_{i \in S} f_i$. We denote by $O_{k\text{-FL}}$ an optimal k-subset of facilities and its objective value by $\text{OPT}_{k\text{-FL}}$.

We remark that the above definition of EXACT k-FACILITY LOCATION is slightly different from the standard definition of k-FACILITY LOCATION used in the literature (Charikar and Li, 2012; Jain and Vazirani, 2001; Zhang, 2007). In the standard version we only require for a feasible solution to have *at most*, rather than *exactly*, k facilities. To the best of our knowledge the exact version of k-FL has not been considered in the literature. In the end of this section, we describe how two known approximation algorithms for standard k-FL can be modified for the exact version. Specifically, we show that there is an approximation algorithm with ratio $\alpha = 3.25 + \epsilon$ for EXACT k-FACILITY LOCATION for any constant $\epsilon > 0$. To avoid cumbersome terminology we use the terms k-FACILITY LOCATION or k-FL to refer to the *exact* version of the problem.

Reduction. Our approximation algorithm for REC k-MEDIAN is essentially a Turing-type reduction to k-FL that produces $|F|$ many instances of k-FL, applies any β-approximation algorithm for k-FL with running time, say, t to each of these instances, and outputs the solution with the smallest REC k-MEDIAN objective. More specifically, given an instance (C, F, k, d, λ) for REC k-MEDIAN, we create for every facility $m \in F$ an instance $R_m = (C, F, k, d, f^m)$ of k-FL where we set the opening cost to $f^m_i = (k-1)\lambda d(i, m)$ for every $i \in F$. We compute an β-approximate solution S_m to each k-FL instance R_m, for $m \in F$, under the k-FL objective. Finally, we output the best of the solutions S_m, for $m \in F$, under the REC k-MEDIAN objective. By analyzing the approximation performance of this reduction in the rest of this section, we prove the following result.

Theorem 2.3. *Any β-approximation algorithm for* EXACT k-FACILITY LOCATION *with running time t can be turned into an 2β-approximation algorithm for* REC k-MEDIAN *with running time $O(|F|t)$. In particular, there is an approximation algorithm for* REC k-MEDIAN *with approximation ratio $2\,\alpha = 6.5 + \epsilon$ with running time $\tilde{\mathcal{O}}\big(k^3|F|(|F|^2 + |C|^2)/\epsilon^2\big)$, for any constant $\epsilon > 0$.*

Analysis. To analyze the approximation performance of the above reduction, let O_{RC} be an optimal solution for the REC k-MEDIAN instance and let the anchor i^* be the 1-median of the set O_{RC}. That is, $i^* = \arg\min_{i \in O_{\text{RC}}} \sum_{i' \in O_{\text{RC}}} d(i, i')$ minimizes the sum of distances to the other facilities in O_{RC}.

Consider the iteration of the algorithm where $m = i^*$. We argue below that in this iteration we get an 2β-approximate solution S_m, which finishes the argument since we output the best solution under the REC k-MEDIAN objective.

In this particular iteration we set the opening cost to $f_i \coloneqq \lambda(k-1)d(i, i^*)$, for each $i \in F$. Let $O_{k\text{-FL}}$ denote an optimal solution for this instance of cost $\text{OPT}_{k\text{-FL}}$.

The following lemma bounds the cost of the algorithm against the cost of an optimal solution to the k-FL instance.

Lemma 2.4. *The cost* $\text{cost}_{\text{RC}}(S_{i^*})$ *of the solution S_{i^*} is at most* $\beta\,\text{OPT}_{k\text{-FL}}$.

Proof. We have

$$\begin{aligned}\text{cost}_{\text{RC}}(S_{i^*}) &= \sum_{j\in C} d(j,S_{i^*}) + \frac{1}{2}\lambda \sum_{i,i'\in S_{i^*}} d(i,i') \\ &\le \sum_{j\in C} d(j,S_{i^*}) + \frac{1}{2}\lambda \sum_{i,i'\in S_{i^*}} (d(i,i^*)+d(i',i^*)) \\ &= \sum_{j\in C} d(j,S_{i^*}) + \lambda(k-1)\sum_{i\in S_{i^*}} d(i,i^*) \\ &= \text{cost}_{k\text{-FL}}(S_{i^*}) \le \beta\,\text{OPT}_{k\text{-FL}}.\end{aligned}$$

□

We complete the proof by bounding the cost of the optimal k-facility location in that problem instance.

Lemma 2.5. *Let* $\text{OPT}_{k\text{-FL}}$ *be cost of the optimal* k*-facility location that corresponds to the instance* R_{i^*}*, and let* OPT_{RC} *be the cost of the optimal reconciliation* k*-median. It is* $\text{OPT}_{k\text{-FL}} \le 2\,\text{OPT}_{\text{RC}}$.

Proof. For each $i \in O_{\text{RC}}$ let $C_i = \sum_{i'\in O_{\text{RC}}\setminus\{i\}} d(i,i')$ denote the connection cost of i to all other facilities in O_{RC}. We have

$$\begin{aligned}\text{OPT}_{k\text{-FL}} &\le \text{cost}_{k\text{-FL}}(O_{\text{RC}}) \\ &= \sum_{j\in C} d(j,O_{\text{RC}}) + \lambda(k-1)\sum_{i\in O_{\text{RC}}} d(i,i^*) \\ &\le 2\sum_{j\in C} d(j,O_{\text{RC}}) + \lambda k C_{i^*} \\ &\overset{(*)}{\le} 2\sum_{j\in C} d(j,O_{\text{RC}}) + \lambda \sum_{i\in O_{\text{RC}}} C_i \\ &\le 2\sum_{j\in C} d(j,O_{\text{RC}}) + \lambda \sum_{i,i'\in O_{\text{RC}}} d(i,i') \\ &= 2\,\text{OPT}_{\text{RC}},\end{aligned}$$

where inequality (*) holds due to the fact that the anchor i^* is the 1-median of the facilities in O_{RC}. □

Approximating Exact k-Facility Location. It remains to provide a constant-factor approximation for EXACT k-FL. Note that in the standard variant of k-FL there are input instances where each optimum solution opens strictly less than k facilities, for example, if the opening costs of the facilities is sufficiently high as compared to the connection cost. Hence, it is not obvious a priori how known algorithms for the standard version can be leveraged for the exact version. This is in contrast to the classic k-median problem where without loss of generality exactly k facilities are opened by some optimal solution. In the appendix, we demonstrate, however, that two known algorithms for standard k-FL can be modified to give the same ratio also the exact version.

The first algorithm has an approximation ratio $\alpha = 3.25 + \epsilon$ and is based on rounding a linear programming relaxation for standard k-FL (Charikar and Li, 2012). In the appendix, we show how to change the LP relaxation so that precisely k fractional facilities are opened. We then combine the algorithm by Charikar and Li (2012) with a dependent rounding scheme by Gandhi et al. (2006) to ensure that also in the resulting integral solution precisely k facilities are opened. The second algorithm for k-FL is based on local search by Zhang (2007). It uses three local neighborhood operations: one is based on swapping facilities preserving the number of opened facilities and the others are based on opening or closing facilities thereby changing the number of facilities. We show that using only the swapping operation (thereby keeping the number of opened facilities exactly k) gives an approximation algorithm for EXACT k-FL.

Theorem 2.6. *There is an approximation algorithm for* EXACT k-FACILITY LOCATION *based on LP-rounding, which has an approximation ratio of* $\alpha = 3.25+\epsilon$ *for any constant* $\epsilon > 0$ *and running time* $\tilde{\mathcal{O}}\big(k^3(|F|^2+|C|^2)/\epsilon^2\big)$*. Moreover, there is an local-search based approximation algorithm for* EXACT k-FACILITY LOCATION *that has an approximation ratio of* $7+\epsilon$ *for any constant* $\epsilon > 0$ *and running time* $\tilde{\mathcal{O}}\big(|F|^5|C|/\epsilon\big)$.

3 SPEEDUP VIA SPARSIFICATION

In this section, we show how sparsification of the input instance can speed up a class of approximation algorithms for REC k-MEDIAN. Our scheme is based on the *adaptive sampling* approach of Arthur and Vassilvitskii (2007). Our main result is that we can efficiently compute a set $S \subseteq F$ of size $\mathcal{O}(k)$ with the guarantee that S contains a constant-factor approximation to REC k-MEDIAN. Assume we are given an arbitrary β-approximation algorithm for REC k-MEDIAN, ALG_{RC}. Our algorithm (see Algorithm 1) has two phases. In the first phase (lines 1 to 8), it runs an adaptive sampling procedure to select $\mathcal{O}(k)$ facilities from F. In the second phase (line 9 to 13), it enumerates all the facilities $i \in F$, augments the sampled set of the first phase with a ball of k facilities centered at i, and then proceeds to run ALG_{RC} on the set of clients C and the sparsified set of facilities F.

Throughout our algorithm and analysis, we make use of parameters α, γ, Δ, μ, and ρ, all of which are constant real values. With some hindsight, we set them as follows: $\Delta = 10$, $\alpha = 3$, $\gamma = 6\Delta + 2 = 62$, $\rho = 1 - 1/(\gamma - 1) \approx 0.984$, and $\mu = 6$. For the sake of readability, we will use the parameters in our manuscript instead of their values. As in the previous works, there is nothing special about the choice of these constants. Specifically, As long as $\Delta \ge 2 + \varepsilon$ and $\gamma \ge 1 + \varepsilon$ for a small ε, the proofs go through. However, there exists a trade-off; while larger values of Δ and γ mean larger approximation factor, lower values for these parameters dictate a larger number of points to be sampled.

One peculiar aspect of our algorithm is that we perform

Algorithm 1: The Sparsifying Approximation Algorithm

input Set of Clients C, set of facilities F, and parameter k.
output $F' \subseteq F$ with $|F| = k$ such that $\text{cost}_{\text{RC}}(F) \leq \max\{12\Delta + 5, 2\gamma + 1\} \cdot \text{OPT}_{\text{RC}}$ with constant probability.

1: Initialize $S_0 \coloneqq \varnothing$;
2: Initialize $T_0 \coloneqq \varnothing$;
3: **for** $t \coloneqq 1$ to μk **do**
4: Sample $j \in C$ with probability $\propto d(j, T_{t-1})$
5: $T_t \coloneqq T_{t-1} \cup \{j\}$
6: $s \coloneqq$ closest center in F to j
7: $S_t \coloneqq S_{t-1} \cup \{s\}$
8: **end for**
9: **for** $i \in F$ **do**
10: $A \coloneqq \{k-1$ facilities from F closest to $i\}$
11: $S \coloneqq S_{\mu k} \cup \{i\} \cup A$
12: $F' \coloneqq \text{ALG}_{\text{RC}}(C, S, k)$
13: **end for**
14: **return** the solution F' of minimum cost

the distance sampling on the set of clients (Line 1), and then include the facility closest to the sampled client to our output set S. The main theorem of this section can be stated as follows (Due to space constraints, we defer all of the proofs of this section to the Appendix):

Theorem 3.1. *Any β-approximation algorithm for* REC k-MEDIAN *with running time $t(|F|, |C|)$ can be turned via the above sparsification scheme into an $\mathcal{O}(\beta)$-approximation algorithm for* REC k-MEDIAN *with running time $O(|F|t(k, |C|))$.*

This theorem implies the following corollaries.

Corollary 3.2. *Any γ-approximation algorithm for* EXACT k-FACILITY LOCATION *with running time $t'(|F|, |C|)$ can be turned into an $\mathcal{O}(\gamma)$-approximation algorithm for* REC k-MEDIAN *with running time $O(|F|t'(k, |C|))$ thereby improving the running times given in Theorem 2.3.*

Corollary 3.3. *There is an LP-based $O(1)$-approximation algorithm for* REC k-MEDIAN *with running time $\tilde{\mathcal{O}}\big(k^3|F|(k^2 + |C|^2)/\epsilon^2\big)$ for any constant $\epsilon > 0$ and a local-search-based $O(1)$-approximation algorithm with running time $\mathcal{O}\big(k^5|F||C|/\varepsilon\big)$.*

In order to show this theorem, we use the reduction to k-FL we presented in Section 2. Assume that we guess the anchor i^*, the actual 1-MEDIAN of the optimum set O_{RC} of facilities for REC k-MEDIAN in the **for** loop of the second phase. Using this guess, we can then define an instance of k-FL as explained in Section 2. We then prove the set of facilities S that we sample and input into ALG_{RC} contains a feasible constant-factor solution to the k-FL instance.[1] Consequently, we are also able to find a good approximation for REC k-MEDIAN.

Lemma 3.4. *Assuming that $i = i^*$ in line 1 of Algorithm 1, with a constant probability, it samples a set of facilities $S \subseteq F$ (in line 1) of size μk such that there exists $S' \subseteq S$ with $|S'| = k$ and $\text{cost}_{k\text{-FL}}(S') \leq \max\{12\Delta + 5, 2\gamma + 1\} \cdot \text{OPT}_{k\text{-FL}}$.*

Proof of Theorem 3.1 follows from Lemma 3.4 in a straightforward manner.

To prove Lemma 3.4, we follow on a high level the strategy of Aggarwal et al. (2009), in which the authors tackled k-MEANS in a Euclidean metric.[2] We should note that the introduction of the disagreement cost in the objective function makes the analysis more involved and technical. To tackle the new technicalities, we introduce a couple of new ideas, which we will highlight shortly.

We fix some iteration $t \geq 1$ of the first phase of the algorithm. We consider the clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ of the clients into clusters C_i induced by an optimal set $O_{k\text{-FL}}$ of k facilities. We classify the cluster as follows. The set of Good clusters that have a relatively low connection cost with respect to the set of facilities sampled so far (or with respect to the anchor), and the rest that we call Bad.

$$\begin{cases} \mathsf{Good}_t \coloneqq \big\{C_\ell \in \mathcal{C} \;:\; \sum_{j \in C_\ell} d(j, T_{t-1} \cup \{i^*\}) \\ \qquad\qquad\qquad \leq \Delta \cdot \sum_{j \in C_\ell} d(j, O_{k\text{-FL}})\big\} \\ \mathsf{Bad}_t \coloneqq \mathcal{C} \setminus \mathsf{Good}_t \end{cases}$$

The general framework is as follows: we first identify a subset of every cluster of clients C_ℓ that we call core(C_ℓ), with the property that if a point from this core is sampled, then the total connection cost of C_ℓ would drop below a constant factor of what it would incur in the optimum solution (see Definition 3.7). We then show that as long as the connection cost induced by the facility subset T_{t-1} sampled so far is *larger* than a constant factor times $\text{OPT}_{k\text{-FL}}$, we sample a point from the core of a Bad cluster in each iteration with constant probability. This in turn is enough to show that in $\mathcal{O}(k)$ iterations, all Bad clusters are turned into Good.

Compared to the work of Aggarwal et al. (2009), two technical challenges need to be addressed:

1. Disagreement cost. In center-based clustering problems such as k-MEDIAN or k-MEANS, it is sufficient to argue that the connection cost of the sampled solution is within a constant factor of optimal with constant probability. The key challenge in Lemma 3 is to upper bound the disagreement cost. By a more careful argument we prove the existence of

[1] Indeed, we show that S contains a set S' with $|S'| \leq k$. In order to turn S' into a feasible solution for the EXACT k-FACILITY LOCATION, we may need to add some more facilities to it. This is done in Lemma 3.14.

[2] By losing a constant factor in approximation factor, we show how similar results can be applied to our k-FL instance while using only the metric property.

a k-subset of facilities that has simultaneously low connection and low disagreement cost. Here, we crucially exploit that the ball A centered at the anchor i^* is suitable to serve remote clients at low enough connection cost while simultaneously keeping the disagreement minimal.

2. Far core points. For the framework of Aggarwal et al. (2009), it is enough to show that, conditioned on sampling from a Bad cluster, we pick something in the core of the cluster. We need to expand this argument and show that what we pick is a client j from a core, but at the same time that j is not too far from the anchor i^* to keep the disagreement low.

To tackle the first challenge, we show in Lemma 3.5 how to (existentially) select at most k facilities from the sampled set such that the total reconciliation (connection plus disagreement) cost remains within a constant factor of $\text{OPT}_{k\text{-FL}}$, if at an iteration t of the sampling we have a relatively low connection cost. To address the second issue, for every cluster of clients C_ℓ induced by the optimal solution, we split $\text{core}(C_\ell)$ into two parts, $\text{near\_core}(C_\ell)$ and $\text{far\_core}(C_\ell)$, with the property that every point in the $\text{near\_core}(C_\ell)$ has a low disagreement cost. We further prove that, conditioned on sampling from a Bad cluster C_ℓ, we pick a facility from $\text{near\_core}(C_\ell)$ with constant probability (see Corollary 3.11 and Lemma 3.12).

3.1 Sparsification via Adaptive Sampling

The first observation about our sampling procedure addresses the disagreement cost issue mentioned above.

Lemma 3.5. *If $\sum_{j\in C} d(j,T_t) \le 2\Delta \cdot \sum_{j\in C} d(j,O_{k\text{-FL}})$ at any round $t \in [\mu k]$ of the algorithm, then there exists a set $S' \subseteq S_t \cup \{i^*\}$ with $|S'| \le k$ such that $\text{cost}_{\text{RC}}(S') \le (12\Delta+5)\sum_{j\in C} d(j,O_{k\text{-FL}}) + 2\lambda k \sum_{i\in O_{k\text{-FL}}} d(i,i^*)$.*

In the remainder of this section, we can assume that $\sum_{j\in C} d(j,T_t) > 2\Delta \cdot \sum_{j\in C} d(j,O_{k\text{-FL}})$. We show that in this case, Algorithm 1 chooses a client from a Bad cluster in each iteration with constant probability.

Lemma 3.6. *If the connection cost to the sampled clients in T_t is large for an iteration t, namely if $\sum_{j\in C} d(j,T_t) > 2\Delta \cdot \sum_{j\in C} d(j,O_{k\text{-FL}})$, then we sample a client from a Bad cluster in Line 1 of Algorithm 1 with probability at least $\frac{1}{2}$.*

Hereinafter, we condition on the event that a client from a Bad cluster C_ℓ is sampled. First, let us precisely define the concept of *core*.

Definition 3.7. For a cluster of clients C_ℓ centered around a facility $i_\ell \in O_{k\text{-FL}}$ and a positive real $\alpha < \Delta - 1$, the *core* is defined as $\text{core}(C_\ell) := \{j \in C_\ell : d(j,i_\ell) \le \alpha \cdot r_\ell\}$, where r_ℓ is the average connection cost of the clients of C_ℓ in the optimal solution, namely $r_\ell := \frac{\sum_{j\in C_\ell} d(j,i_\ell)}{|C_\ell|}$.

We observe that if, in an iteration t, first a client from $\text{core}(C_\ell)$ is chosen for $C_\ell \in \text{Bad}_t$ and next the closest facility to this client, say i'_ℓ, is added to S_t, then we get:

Observation 1. $\sum_{j\in C_\ell} d(j,i'_\ell) \le (2\alpha+1)\sum_{j\in C_\ell} d(j,i_\ell)$.

In other words, sampling from core of C_ℓ will turn it from Bad to Good. We also will make use of the following observation for a Bad cluster C_ℓ during iteration t:

Observation 2. *Let $\delta_{\ell,t} := d(i_\ell, y)$, where $y := \arg\min_{j\in T_{t-1}}\{d(i_\ell,j)\}$ is the closest sampled client to i_ℓ before iteration t. Then, $\delta_{\ell,t} \ge (\Delta-1)r_\ell$.*

The next two lemmas show that, in any iteration t of the sampling algorithm, a good fraction of the connection cost of every Bad cluster C_ℓ (with respect to T_t) comes from $\text{core}(C_\ell)$. Therefore, with a constant probability, we sample a client from the core of a Bad cluster, conditioned on sampling from Bad. First, we show that the size of the core is considerably large.

Lemma 3.8. *For all $C_\ell \in \mathcal{C}$, $|\text{core}(C_\ell)| \ge \left(1-\frac{1}{\alpha}\right)|C_\ell|$.*

Lemma 3.9.
$\mathbb{P}\left[\text{sample } j \in \text{core}(C_\ell) \mid j \in C_\ell \text{ and } C_\ell \in \textit{Bad}_t\right] \ge (1-\frac{1}{\alpha})(1-\frac{\alpha+1}{\Delta})$.

At this point, we guarantee that as long as we are sampling clients from Bad clusters, we hit their core with a constant probability. Here is when we face the second issue, which is how to also guarantee that the clients we hit from the core are not too far from the anchor i^*. We need this property to keep the disagreement cost within constant factor of that of the optimum solution. To tackle this, we introduce the notions of *far* and *near* core.

Definition 3.10. For a cluster C_ℓ of clients, we define $\text{far\_core}(C_\ell) := \{j \in \text{core}(C_\ell) : d(j,i^*) > \gamma \cdot d(i_\ell,i^*)\}$, where i_ℓ is the optimum facility for the cluster C_ℓ. We also define $\text{near\_core}(C_\ell) := \text{core}(C_\ell) \setminus \text{far\_core}(C_\ell)$.

Next, we observe that for a cluster $C_\ell \in \text{Bad}_t$ where the near core is large enough, we hit this core with constant probability. It is straightforward to see this statement as a corollary of Lemma 3.9:

Corollary 3.11. *For a cluster of clients $C_\ell \in \textit{Bad}_t$, if $|\text{near\_core}(C_\ell)| \ge \rho \cdot |\text{core}(C_\ell)|$, then*
$\mathbb{P}\left[j \in \text{core}(C_\ell) \mid j \in C_\ell \text{ and } C_\ell \in \textit{Bad}_t\right] \ge \rho \cdot \left(1-\frac{1}{\alpha}\right)\left(1-\frac{\alpha+1}{\Delta}\right)$.

Note that if we sample a client from the near core in T_t, we then proceed to pick the closest facility to this client and add it to S_t. Let i' denote this facility. We show that the disagreement cost incurred by this facility is within a constant factor of the disagreement cost of i_ℓ, the optimum facility for C_ℓ.

Observation 3. $d(i',i^*) \le (2\gamma+1)\cdot d(i_\ell,i^*)$.

So, it remains to argue for the case where $|\text{far\_core}(C_\ell)| \ge (1-\rho)\cdot|\text{core}(C_\ell)|$ for a cluster C_ℓ. In such a scenario,

we prove that $C_\ell \in \mathsf{Good}_t$ by showing that the anchor i^* already lies within the radius of the near core of C_ℓ. This implies that if we redirect all the clients in C_ℓ to i^*, we incur a connection cost of at most $\alpha \cdot \sum_{j \in C_\ell} d(j, i_\ell)$ and a disagreement cost of $0 = d(i^*, i^*)$.

Lemma 3.12. *If* $|\text{far\_core}(C_\ell)| \geq (1-\rho) \cdot |\text{core}(C_\ell)|$ *for a cluster* C_ℓ *then* $d(i^*, i_\ell) \leq \alpha \cdot r_\ell$.

Putting everything together, we get the following lemma.

Lemma 3.13. *Suppose Algorithm 1 samples a client* j' *from the cluster* C_ℓ *at iteration* t. *Then*
$\mathbb{P}\left[\sum_{j \in C_\ell} d(j, j') \leq (2\alpha+1) \sum_{j \in C_\ell} d(j, \sigma^*(j))\right]$
$\geq \frac{\rho}{2} \cdot \left(1 - \frac{1}{\alpha}\right)\left(1 - \frac{\alpha+1}{\Delta}\right)$.

At this point, we can use the proof of Theorem 1 from Aggarwal et al. (2009) in a black-box manner (although with different constants) to prove Lemma 3.4 via modeling the evolution of the clusters from Bad to Good as a super-martingale sequence of random variables.

3.2 Padding the Sampled Set

In this section, we show how to compensate for the deficit in cardinality of the sampled set without increasing the total cost by much:

Lemma 3.14. *If for the sampled set of facilities* S *we have that* $|S| < k$, *then we have that* $\text{cost}_{\text{RC}}(S \cup A) \leq \sum_{j \in C} d(j, S) + \sum_{i \in S} \lambda k \cdot d(i, i^*) + \gamma \cdot \sum_{i \in O_{k\text{-FL}}} \lambda k \cdot d(i, i^*)$, *where* $A = \{\max\{0, k - |S|\}$ *facilities from* F *closest to* $i^*\}$.

Note that, if the cost of S is a constant approximation of $\text{OPT}_{k\text{-FL}}$, then the cost of the padded set $S \cup A$ is also withing a constant factor of $\text{OPT}_{k\text{-FL}}$.

3.3 Unbounded Gap for the Vanilla Sampling

In this section, we justify our sampling procedure by showing that the sampling procedure of Arthur and Vassilvitskii renders insufficient in our setting.

Claim 1. *The adaptive sampling of Arthur and Vassilvitskii can produce a sample set* S *of* $\mathcal{O}(k)$ *clients such that any subset* $S' \subseteq S$ *with* $|S'| = k$ *would incur a cost* $\text{cost}_{k\text{-FL}} S' = \Omega(\lambda k \cdot \text{OPT}_{k\text{-FL}})$ *where* λ *and* k *are part of the input.*

Proof. Consider the instance in Figure 1. In this instance, we assume $C = F$, with $|C| = n$. As a result, we refer to the clients/facilities as simply *points* in the following. There are μk points on the outer ring, $k-1$ points in the middle ring, and $n - (\mu+1)k + 1$ points in the inner circle, where the anchor point i^* is located. Let the distance between every pair of points in the circle be a negligible ε. Also, for an arbitrarily large M allow the distance between every

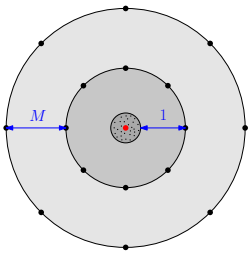


Figure 1: Instance with unbounded cost for the vanilla adaptive sampling

point in the outer ring to a point in the middle ring be M, and let the distance between any point in the middle ring to a point in the inner circle be 1. It is straightforward to see that for large enough values of M, the sampling procedure only picks from the outer ring in the sample set S in addition to the anchor i^*. As a result, the disagreement cost incurred by any subset $S' \subseteq S$ with cardinality k is roughly $\approx \lambda k^2 \cdot M$ while the connection cost would be $\mathcal{O}(\mu k \cdot M)$. On the other hand, the optimum solution to the k-FL instance will choose the anchor together with the $k-1$ points from the middle ring, incurring disagreement and connection costs of $\mathcal{O}\big(\lambda k^2\big)$ and $\mathcal{O}(\mu k \cdot M)$, respectively. More precisely,

$$\frac{\text{cost}_{k\text{-FL}} S'}{\text{OPT}_{k\text{-FL}}} = \Big[\lambda k^2 \cdot (M + 1 + \varepsilon) + (k-1)(1-\varepsilon) + (\mu - 1)k \cdot (M + 1 + \varepsilon)\Big] / \Big[\lambda k^2 + (k-1) \cdot (M + 1 + \varepsilon)\Big],$$

which tends to λk as M grows towards infinity. □

Finally, we make a relatively straightforward comment regarding the reason for sampling from the client set and not from the facility set. The main intuition behind the sampling procedure is that as long as the connection cost is large, it is reasonable to randomly pick a client from far groups of clients, and hopefully open a facility near them in order to reduce the cost. If one chooses to sample directly from the facilities, an adversary can easily design an instance where there are large groups of facilities very far from any client. This would trick the sampling algorithm to open a facility from such "useless" groups which can lead to unbounded connection and total cost.

4 EMPIRICAL EVALUATION

We conduct experiments to evaluate the performance of the proposed methods. We seek to answer the two following questions: (1) is the proposed constant-factor approximation algorithm superior to the local-search heuristic in practice? (2) does the proposed sparsification scheme allow us to significantly reduce computation times without compromising the quality of the obtained solution?

Table 1: Characteristics of the datasets.

Name	Number of clients	Number of facilities
CONGRESS	420	420
WIKIELECTIONS	7 115	200

We evaluate the following three methods: (1) LOCALSEARCH: the known local-search heuristic (Ordozgoiti and Gionis, 2019). (2) k-UFL+LP: our proposed reduction to k-FL, using our adaptation of the LP rounding algorithm of (Charikar and Li, 2012) to solve each of the resulting k-FL instances. (3) k-UFL+LS: our proposed reduction to k-FL, using our adapted local-search heuristic to solve each of the resulting k-FL instances. We use our own C++ implementation of all algorithms. Our setup is described in the supplement.

We consider two datasets, corresponding to use cases suited to the REC k-MEDIAN problem.

CONGRESS:[3] voting records of the 115th US Congress. Each data point corresponds to a congressperson, and each variable represents whether they voted positively or negatively for a particular bill. For details, see (Ordozgoiti and Gionis, 2019). We consider all congresspeople to be clients and candidate facilities, i.e., $F = C$.

WIKIELECTIONS:[4] Signed graph data representing Wikipedia moderator elections (Leskovec et al., 2010). Each vertex corresponds to a user. An edge from user i to user j exists if the former voted for the latter in a moderator election. The sign of the edge indicates whether the vote was positive or not. We ignore edge directions. For each vertex we compute the spectral embedding corresponding to the bottom 25 eigenvectors of the normalized signed Laplacian (Kunegis et al., 2010).[5] The facility set F is comprised of the 200 users with largest degree. We compute Euclidean distances between the resulting embedded vertices.

Performance Evaluation. We evaluate the performance of the methods on the CONGRESS and WIKIELECTIONS datasets, for different choices of k and λ. We limit k-UFL+LP to CONGRESS dataset, given the computational cost of solving the linear program. We measure the average cost of the solution found by each of them over 20 runs, and report the ratio of the cost achieved by LOCALSEARCH to the cost achieved by k-UFL+LP and k-UFL+LS. The results are shown in Figure 2. Even though all methods find solutions of similar quality, LOCALSEARCH does so less reliably than k-UFL+LS and k-UFL+LS for larger values of k and λ. Even when these parameters grow, k-UFL+LP and k-UFL+LS consistently find the same solution, whereas

[3] https://zenodo.org/record/2573954

[4] https://snap.stanford.edu/data/wiki-Vote.html

[5] The dimensionality of the embedding was chosen by visual inspection of the spectrum.

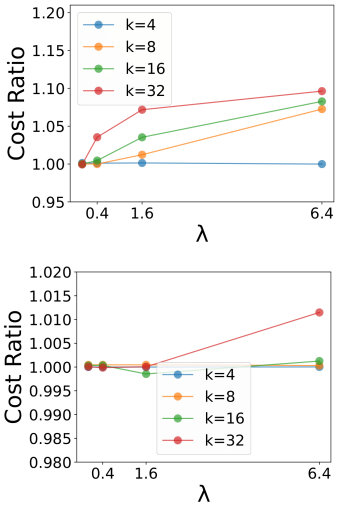


Figure 2: Performance of the different algorithms on the CONGRESS (top) and WIKIELECTIONS (bottom) datasets. We report the ratio of the cost yielded by LOCALSEARCH to that achieved by k-UFL+LP and k-UFL+LS. Since the latter two consistently yield the same solution, we plot a single line.

LOCALSEARCH is sensitive to the initial set and occasionally produces solutions of poorer quality. This is reflected by the growing ratios.

We report average running times in Table 2. We omit k-UFL+LP due to its impractical running times. This shows that the improvements in quality come at a noticeable cost in running times.

Sparsification. We assess the effectiveness of the proposed sparsification approach as a method for speeding up algorithms for REC k-MEDIAN. We consider the WIKIELECTIONS dataset. We fix $k = 8, \lambda = 1.6$, and vary the value of μ (see Algorithm 1) so that the number of sampled clients is in $\{3, 6, 12, 25, 50, 100, 200\}$. Note that each of these values will result in a certain number of sampled facilities, among which the algorithms will choose the final set. The results show how the method allows us to significantly reduce computation time without sacrificing quality. Due to the space limitations, the corresponding figures are in the supplement.

4.1 Experiments on Synthetic Data

In order to improve our understanding of the behaviour of the methods under scrutiny, we conduct experiments on synthetic datasets. Our goal is to test the robustness of the methods proposed in this paper — k-UFL+LS in particular — compared to LOCALSEARCH.

For both methods, we will vary one parameter which allows us to trade computation time for solution quality. In the

Table 2: Running times of LocalSearch and k-UFL+LS on the different datasets.

	Congress				WikiElections			
	$k=4$	$k=8$	$k=16$	$k=32$	$k=4$	$k=8$	$k=16$	$k=32$
LocalSearch	0.10	0.26	0.65	2.14	3.68	9.55	22.45	56.35
k-UFL+LS	13.22	35.44	94.46	245.80	64.98	165.48	322.72	741.58

case of LocalSearch, we will allow more or less initial solutions, sampled uniformly at random. For k-UFL+LS, we will consider more or less facilities in the "guessing" phase, where we consider different facilities as the candidate 1-median of the optimal set. In the plots, we will refer to both these parameters as *Sample size*.

We generate two datasets containing 100 facilities in $\mathbb{R}^2$. In the first one, all clients located within a ball of small radius r_A centered at the origin, along with a small cluster of 5 facilities within a ball of slightly larger radius r_C. We place the remaining 95 facilities roughly at a distance of $r_B \gg r_A$ from the origin. The performance of LocalSearch will be determined entirely by whether or not the starting solution contains enough of the 5 well-situated facilities. When the solution is mostly comprised of far-away facilities, the "reconciliation" term of the objective will prevent the algorithm from exploring the ones near the clients. For k-UFL+LS, on the contrary, it will be sufficient to sample one candidate 1-median in this set. We choose the values $r_A = 1$, $r_C = 2$, and $r_B = 100$. We set $k = 5$ and $\lambda = 1$.

In Figure 3 (left), we show the estimated probabilities that these events will take place. For each *Sample size* we run both algorithms 200 times and count the fraction of times that a favourable event occurs. While k-UFL+LS requires sampling about 40 facilities to provide a reasonable chance of success, LocalSearch struggles to sample a good initial subset even with an allowance of 100 attempts. We note that the running time both methods is roughly equal when the *Sample size* is the same.

We emphasize that this is not always the case in datasets presenting similar structure. We modify the previously described dataset by placing the 95 far-away facilities along the circle of radius r_B. In this case, the reconciliation term no longer incurs a disproportionate penalty for sensible swaps. As expected, k-UFL+LS performs just like before, but the structure of these data clearly benefits LocalSearch.

5 CONCLUSION, LIMITATIONS, AND FUTURE WORK

In this paper, we proposed a constant-factor approximation algorithm for reconciliation k-median, improving significantly the previous results on this problem. Our algorithm relies on a connection between reconciliation k-median and a variant of the k-facility location problem. In addition,

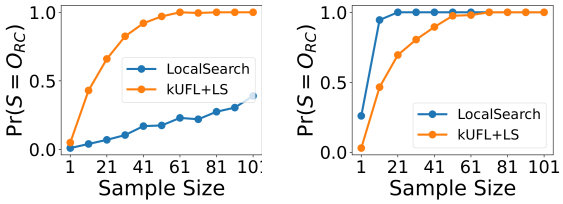


Figure 3: Experiments on synthetic data.

we develop a sparsification scheme based on adaptive sampling, which allows to reduce the number of potential cluster centers to $\mathcal{O}(k)$ and providing significant speedup.

Our empirical evaluation illustrates the improved performance and stability of the proposed approach, albeit at an increased cost in computational cost. Nevertheless, we have illustrated how our sparsification scheme can be leveraged to significantly decrease running times.

Despite the significant speedup obtained with our sparsification procedure, our method relies on repeated calls to k-facility location instances, and thus, its practical performance is limited by the available methods for the latter problem. Developing truly scalable methods that go beyond this limitation is a challenge left for future work. Another future direction is to design improved methods for special metrics, e.g., the Euclidean distance.

Acknowledgements. Aristides Gionis is supported by the Academy of Finland project MLDB (325117), the ERC Advanced Grant REBOUND (834862), the EC H2020 RIA project SoBigData++ (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Kamyar Khodamoradi was supported by Deutsche Forschungsgemeinschaft - Project number 399223600.

References

Aggarwal, A., Deshpande, A., and Kannan, R. (2009). Adaptive sampling for k-means clustering. In Dinur, I., Jansen, K., Naor, J., and Rolim, J. D. P., editors, *Proc. 12th International Workshop on Approximation, Randomization, and Combinatorial Optimization (APPROX'09)*, volume 5687 of *Lecture Notes in Computer Science*, pages 15–28. Springer.

Ailon, N., Jaiswal, R., and Monteleoni, C. (2009). Stream-

ing k-means approximation. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, pages 10–18. Curran Associates, Inc.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In Bansal, N., Pruhs, K., and Stein, C., editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM.

Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. (2004). Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562.

Ben-David, S. (2007). A framework for statistical clustering with constant time approximation algorithms for k-median and k-means clustering. *Machine Learning*, 66(2):243–257.

Byrka, J., Pensyl, T., Rybicki, B., Spoerhase, J., Srinivasan, A., and Trinh, K. (2018). An improved approximation algorithm for knapsack median using sparsification. *Algorithmica*, 80(4):1093–1114.

Charikar, M., Guha, S., Tardos, É., and Shmoys, D. B. (2002). A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149.

Charikar, M. and Li, S. (2012). A dependent LP-rounding approach for the k-median problem. In Czumaj, A., Mehlhorn, K., Pitts, A. M., and Wattenhofer, R., editors, *Proc. 39th International Colloquium on Automata, Languages, and Programming (ICALP'12)*, volume 7391 of *Lecture Notes in Computer Science*, pages 194–205. Springer.

Chen, K. (2008). A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 826–835.

Chierichetti, F., Kumar, R., Lattanzi, S., and Vassilvitskii, S. (2017). Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*.

Demirci, H. G. and Li, S. (2016). Constant approximation for capacitated k-median with $(1+\epsilon)$-capacity violation. In Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., and Sangiorgi, D., editors, *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP'16)*, volume 55 of *LIPIcs*, pages 73:1–73:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Eisenbrand, F., Grandoni, F., Rothvoß, T., and Schäfer, G. (2008). Approximating connected facility location problems via random facility sampling and core detouring. In Teng, S., editor, *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, San Francisco, California, USA, January 20-22, 2008*, pages 1174–1183. SIAM.

Feldman, D., Monemizadeh, M., and Sohler, C. (2007). A PTAS for k-means clustering based on weak coresets. In Erickson, J., editor, *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 11–18. ACM.

Gandhi, R., Khuller, S., Parthasarathy, S., and Srinivasan, A. (2006). Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360.

Garey, M. R. and Johnson, D. S. (1979). Computers and intractability. *A Guide to the*.

Han, L., Wu, C., and Xu, Y. (2021). Approximate the lower-bounded connected facility location problem. In Chen, C., Hon, W., Hung, L., and Lee, C., editors, *Computing and Combinatorics - 27th International Conference, COCOON 2021, Tainan, Taiwan, October 24-26, 2021, Proceedings*, volume 13025 of *Lecture Notes in Computer Science*, pages 487–498. Springer.

Jain, K. and Vazirani, V. V. (2001). Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM (JACM)*, 48(2):274–296.

Kunegis, J., Schmidt, S., Lommatzsch, A., Lerner, J., De Luca, E. W., and Albayrak, S. (2010). Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 559–570. SIAM.

Langberg, M. and Schulman, L. J. (2010). Universal epsilon-approximators for integrals. In Charikar, M., editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 598–607. SIAM.

Leskovec, J., Huttenlocher, D., and Kleinberg, J. (2010). Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1361–1370.

Li, S. and Svensson, O. (2016). Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547.

Ordozgoiti, B. and Gionis, A. (2019). Reconciliation k-median: Clustering with non-polarized representatives. In *The World Wide Web Conference*, pages 1387–1397.

Srinivasan, A. (2001). Distributions on level-sets with applications to approximation algorithms. In *Proc. 42nd Annual Symposium on Foundations of Computer Science (FOCS'01)*, pages 588–597. IEEE Computer Society.

Zhang, P. (2007). A new approximation algorithm for the k-facility location problem. *Theoretical Computer Science*,

384(1):126–135. Theory and Applications of Models of Computation.

A APPROXIMATING EXACT k-FACILITY LOCATION

A.1 LP-Based Approximation Algorithm

We modify the algorithm by Charikar and Li (2012) for the standard version of k-FL to give an algorithm with the same ratio also for EXACT k-FL. There are three adjustments (described below in detail) to algorithm or analysis. First, we modify the LP relaxation so that precisely k fractional facilities are opened (rather than $\leq k$ facilities). Second, we employ the dependent rounding scheme by Gandhi et al. (2006) (rather than the one by Srinivasan (2001) as in the original algorithm by Charikar and Li). This is done in a way so that precisely k (integral) facilities are opened in the rounded solution and so that essential properties (preservation of marginals, negative correlation) needed in the analysis are still preserved. The third adjustment is that the algorithm by Charikar and Li creates multiple copies of the initial facilities to achieve certain nice properties in the fractional solution. We need to ensure that in the dependent rounding at most one copy of each facility is opened. While this was irrelevant for the standard version, it becomes necessary for the exact version of k-FL.

A.1.1 The *k*-UFL Algorithm by Charikar and Li

First, we recap the algorithm by Charikar and Li (2012) for the standard version of k-FL. Later, we modify it to return exactly k facilities. The algorithm gives a 3.25-approximation for k-FL in expectation and is subdivided into the following steps, which we outline here:

1. **Linear program (LP):** First, the below standard linear programming relaxation for k-FL is solved. For each facility i we introduce a (fractional) indicator variable y_i representing the extent to which i is opened. For any client j, variable $x_{i,j}$ indicates how much of the demand of client j is served by i.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i\in F, j\in C} d(i,j)x_{j,i} + \sum_{i\in F} f_i y_i \\
\text{subject to} \quad & \sum_{i\in F} x_{i,j} = 1, \quad \forall j \in C, \\
& \sum_{i\in F} y_i \leq k, \\
& x_{i,j} \leq y_i, \quad \forall i \in F, j \in C, \\
& x_{i,j}, y_i \in [0,1], \forall i \in F, j \in C
\end{aligned}$$

2. **Splitting phase:** In this phase, we modify the input instance and the LP solution so that $x_{i,j} = y_i$ for every client $j \in C$ and every facility $i \in F$. We achieve this by splitting some of the facilities into multiple identical copies of that facility. If $i \in F$ is split into $t \geq 2$ many facilities then it is called a *split facility* and the t many facilities $\{i.1, i.2, \ldots, i.t\}$ that were created by this split, are called *fractional facilities*. After the split, $y_i = y_{i.1} + y_{i.2} + \cdots + y_{i.t}$, which means that the total volume $\sum_i y_i$ of opened facilities is preserved. From now on we call the original set of facilities F_{old} and the new set of facilities F_{new}.

3. **Filtering phase:** In this phase a subset $C' \subseteq C$ is determined, where $j, j' \in C'$ are sufficiently far apart from each other. (We do not need the precise requirement on C' for our below argument.)

4. **Bundling phase:** Here we assign to every client $j \in C'$ a set of facilities $U_j \subseteq F_{\text{new}}$, where $i \in U_j$ is close enough to j and i serves j. This leads to $\frac{1}{2} \leq \text{vol}(U_j) \leq 1$, with $\text{vol}(U_j) = \sum_{i\in U_j} y_i$. In this procedure facilities can stay unbundled.

5. **Matching phase:** We match the closest pair $j, j' \in C'$ of clients, add the pair to the set M, and iterate with the remaining clients in C'. In doing so, one $b \in C'$ can stay unmatched.

6. **Sampling phase:** Finally, we open a subset of the facilities using a dependent rounding procedure described below.

We remark that we specify the above steps only to the extent needed for our below arguments.

We introduce the following set of operations.

1. **Opening/closing a facility i:** Opening a facility i means to add i to the solution. Closing a facility i means that it will never be added to the solution.
2. **Opening/closing a bundle j:** To open a bundle U_j, $j \in C'$ means to randomly open exactly one facility $i \in U_j$, with the probability $y_i/\text{vol}(U_j)$. Closing a bundle j means that every facility in U_j is closed.
3. **Opening/closing a matched pair (j, j'):** Opening a pair $(j, j') \in M$ means to open both bundles j and j'. Closing a pair (j, j') is equivalent to open either bundle j or j'. When the matched pair is closed, j is opened with probability $1 - \text{vol}(U_{j'})$ and j' is opened with probability $1 - \text{vol}(U_j)$.

If a facility, bundle or pair is not opened, then it is closed.

In a naive implementation of the sampling phase every $(j, j') \in M$ would be opened *independently* with probability $\text{vol}(U_j) + \text{vol}(U_{j'}) - 1$, the unmatched bundle b would be opened randomly with probability $\text{vol}(U_b)$ and every unbundled facility i with probability y_i. Notice that this results in:

$$\forall j \in C' \colon \mathbb{P}\left[\text{“}U_j \text{ is open”}\right] = \text{vol}(U_j)$$
$$\forall i \in F_{\text{new}} \colon \mathbb{P}\left[\text{“}i \text{ is open”}\right] = y_i \,.$$

As a result the *expected* number of opened facilities from F_{new} would be at most k. Of course, this is not sufficient to guarantee feasibility. The authors argue, that the naive implementation satisfies

$$\mathbb{E}\left[\text{cost}_{k\text{-FL}}(S)\right] \leq 3.25 \left(\sum_{i \in F, j \in C} d(i,j) x_{j,i} + \sum_{i \in F} f_i y_i \right)$$

and hence gives a 3.25-approximate pseudo-solution in expectation.

To additionally ensure feasibility, that is, opening at most k facilities, Charikar and Li resort to a *dependent* rounding procedure to *always* return at most k facilities from F_{new}. Let $F_{\text{unbundled}}$, $F_{\text{unmatched}}$ and A be defined as follows:

$$F_{\text{unbundled}} = \{i : i \text{ is an unbundled facility}\}$$
$$F_{\text{unmatched}} = \{b : U_b \text{ is an unmatched bundle}\}$$
$$A = F_{\text{unbundled}} \cup F_{\text{unmatched}} \cup M$$

First the authors define a random variable X_a and a variable x_a for every $a \in A$.

$$X_a = \begin{cases} 1, & \text{if } a \text{ is open} \\ 0, & \text{if } a \text{ is not open} \end{cases}$$

$$x_a = \begin{cases} y_a, & \text{if } a \text{ is an unbundled facility} \\ \text{vol}(U_a), & \text{if } a \text{ is an unmatched bundle} \\ \text{vol}(U_j) + \text{vol}(U_{j'}) - 1, & \text{if } a = (j, j') \in M \end{cases}$$

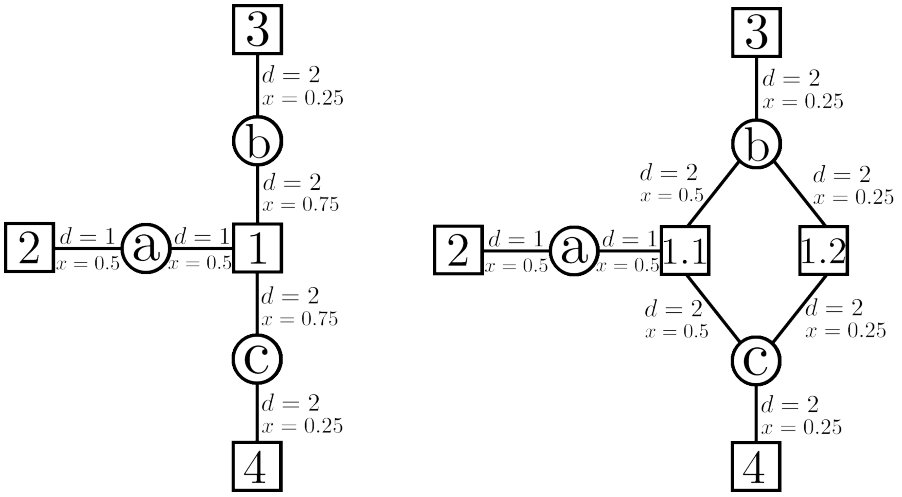
This implies $\sum_{a \in A} x_a \leq k - |M|$. Hence, if the rounding preserves the volume, that is, $\sum_{a \in A} X_a = \lceil \sum_{a \in A} x_a \rceil$, then there are at most k open facilities of F_{new}.

To achieve volume preservation, the tree-based dependent rounding procedure of Srinivasan (2001) is employed. It guarantees the following properties:

1. $\mathbb{P}\left[X_a = 1\right] = x_a$.
2. $\mathbb{P}\left[\sum_{a \in A} X_a \leq \lceil \sum_{a \in A} x_a \rceil\right] = 1$
3. The following negative correlation properties hold for all $S \subseteq A$:

$$\mathbb{P}\left[\bigwedge_{a \in S} X_a = 0\right] \leq \prod_{a \in S} \mathbb{P}\left[X_a = 0\right]$$

$$\mathbb{P}\left[\bigwedge_{a \in S} X_a = 1\right] \leq \prod_{a \in S} \mathbb{P}\left[X_a = 1\right]$$



(a) LP solution before the split process

(b) LP solution after the split process

Figure 4: An example with $F = \{1, 2, 3, 4\}$, $C = \{a, b, c\}$, $k = 3$ and the opening cost f for each facility is 1. The distances that are not drawn, are the minimal paths through this graph, e.g. $d(b, a) = 3$, hence the triangle inequality still holds. The x's represent how connected a client is to a facility in the LP solution. The y of a facility in the LP solution is the maximum x adjacent to the facility, e.g. $y_1 = 0.75$.

Note that Property 2 implies that the procedure opens at most k facilities. Charikar and Li argue that in their analysis of the above-described naive, independent rounding scheme only Property (1) and negative correlation is used. Hence also the dependent rounding scheme gives a 3.25-approximate solution in expectation. The dependent rounding scheme can be derandomized with the method of conditional expectations.

A.1.2 Adjustments for EXACT k-FL

We now describe three adjustments we make to the above algorithm by Charikar and Li to also give a 3.25-approximation for EXACT k-FL.

First, we replace the inequality constraint $\sum_{i \in F} y_i \leq k$ in the LP relaxation by an equality constraint $\sum_{i \in F} y_i = k$. Note that, when we solve the resulting LP relaxation for the exact version, we obtain a feasible solution to the original relaxation in particular. This allows us to apply some of the existing arguments in a black box fashion.

Suppose that we run the vanilla rounding algorithm by Charikar and Li on an optimum solution to the modified LP relaxation for Exact k-FL. By the volume preservation property (1) this algorithms opens *precisely* k facilities. Unfortunately, in their procedures two copies of the same split facility may be picked, which would result in effectively strictly less than k facilities from F_{old} to be opened. Figure A.1.2 shows an example, where this actually happens. Since $C' = \{a\}$, $U_a = \{1.1, 2\}$ and the set of unbundled facilities is $\{1.2, 3, 4\}$ this demonstrated that the vanilla rounding can open less than k facilities of F_{old}.

We ensure that the rounding opens precisely k facilities of F_{old} by guaranteeing that for each split facility their copies are in the same bundle. This results in only one fractional facility of a split facility to be opened at most. To achieve this, we leverage the rounding procedure for bipartite graphs of Gandhi et al. (2006) rather than the tree-based rounding by Srinivasan (2001).

The algorithm by Gandhi et al. (2006) receives a bipartite graph (B, C, E) and a list of values $x_{v,u}$ for each edge $(v, u) \in E$ as input. It rounds every $x_{v,u}$ to a random variable $X_{v,u} \in \{0, 1\}$ and returns the resulting set of all $X_{v,u}$. We define the quantities

$$d_v = \sum_{(v,u) \in E} x_{v,u} \text{ and } D_v = \sum_{(v,u) \in E} X_{v,u},$$

where d_v denotes the fractional degree of the vertex v and D_v denotes the integral degree. The dependent rounding procedure by Ghandi et al. ensures the following properties.

(a) $\mathbb{P}[X_{v,u} = 1] = x_{v,u}$.

(b) Degree-preservation. $\mathbb{P}[D_v \in \{\lfloor d_v \rfloor, \lceil d_v \rceil\}]$. Notice, if $d_v \in \mathbb{N}$, then $D_v = d_v$

(c) Negative correlation holds for any $v \in B \cup C$ and for any $S \subseteq \{X_{(v,i)} : (v,i) \in E\}$:

$$\mathbb{P}\left[\bigwedge_{X \in S} X = 0\right] \leq \prod_{X \in S} \mathbb{P}[X = 0]$$

$$\mathbb{P}\left[\bigwedge_{X \in S} X = 1\right] \leq \prod_{X \in S} \mathbb{P}[X = 1]$$

Notice that the bipartite graph procedure can simulate tree-based rounding by using a star graph as follows. We set $B = \{r\}$ where r is the root of the star, $C = A$, and $x_{a,r} = x_a$ where A and x_a are defined as in Section A.1.1.

To ensure that we open exactly k facilities from the original set F_{old} we define a bipartite graph appropriately.

First, a root r has to be added to the graph. Every unbundled facility, unmatched bundle and matched pair is then connected to r with the gadgets described below. See Figure A.1.2 for an illustration of the gadgets.

Unbundled facility: See Figure 5(c). An unbundled facility i is simply attached to r, with $x_{r,i} = y_i$. Therefore it is opened with the probability y_i.

Unmatched bundle: See Figure 5(a). The unmatched bundle b is attached to the root via a vertex μ, with $x_{r,\mu} = \text{vol}(U_b)$. Furthermore a vertex φ is connected to μ, with $x_{\mu,\varphi} = 1 - \text{vol}(U_b)$. Therefore

$$d_\mu = D_\mu = X_{r,\mu} + X_{\mu,\varphi} = 1\,.$$

The edge (μ, φ) can be seen as a negation to (r, μ), since either $X_{r,\mu}$ or $X_{\mu,\varphi}$ is 1, and the other is 0. Attached to φ are all the facilities $b_i \in U_b$, with $x_{\varphi,b_i} = y_{b_i}$. Thus

$$d_\varphi = D_\varphi = 1$$

and at most one X_{φ,b_i} with $b_i \in U_b$ can be equal to 1. Summarizing, our gadget ensures that precisely one facility $b_i \in U_b$ is opened if $X_{r,\mu} = 1$. If $X_{r,\mu} = 0$ then no facility in U_b is opened.

Matched pair: See Figure 5(b). The matched pair (j, j') is represented via a vertex γ connected to the root r. The bundle j is represented by a vertex α connected to γ. Bundle j' is represented by node β connected to γ as well. The edges have the following weights

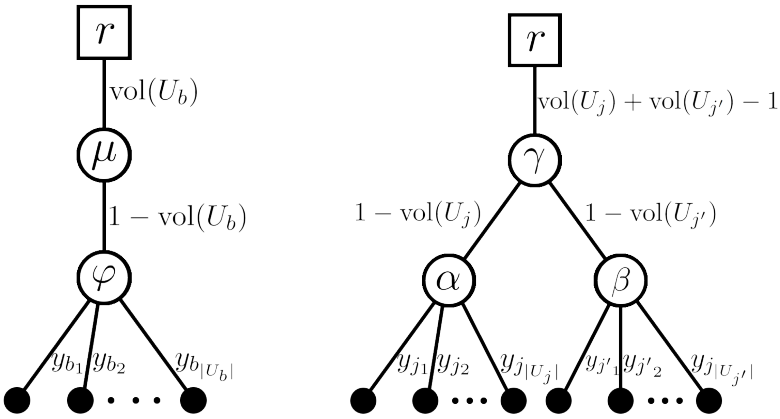
$$\begin{aligned} x_{r,\gamma} &= \text{vol}(U_j) + \text{vol}(U_{j'}) - 1 \\ x_{\gamma,\alpha} &= 1 - \text{vol}(U_{j'}) \\ x_{\gamma,\beta} &= 1 - \text{vol}(U_j)\,. \end{aligned}$$

Similar to the unmatched bundle, every facility $\in U_j$ is attached to α with $x_{\alpha,i} = y_i$ and every facility $i' \in U_{j'}$ is attached to β with $x_{\beta,i'} = y_{i'}$. Since $D_\gamma = 1$, only one of the edges (r,γ), (γ,α) and (γ,β) will be rounded to 1. Here $X_{(r,\gamma)} = 1$ means that both bundles j, j' are opened. The event $X_{(\gamma,\alpha)} = 1$ means that j is closed while j' is opened. Analogously, $X_{(\gamma,\beta)} = 1$ means that j is opened while j' remains closed. Hence the gadget ensure that our modified dependent rounding treats the pair (j, j') as the tree-based rounding.

Buffer vertices: For each copy i of some split facility we introduce two buffer vertices ω_i and ψ_i. Node ω_i is connected to i and ψ_i is connected to ω_i where

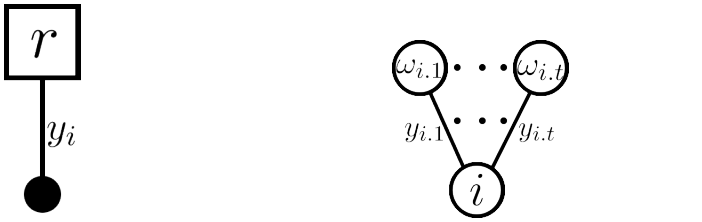
$$\begin{aligned} x_{i,\omega_i} &= 1 - y_i \\ x_{\omega_i,\psi_i} &= y_i \end{aligned}$$

The edge (ω_i, ψ_i) indicates if facility i is open or not.



(a) Unmatched bundle b

(b) Matched bundles (j, j')



(c) Unbundled facility i

(d) Split facility i and it's fractional facilities $i.1, \ldots, i.t$. There are no ψ's, since they fused into i.

Figure 5: The different structures for the bipartite graph.

Split facilities: See Figure 5(d) If $i \in F_{\text{old}}$ is split into copies $\{i.1, i.2, \ldots, i.t\}$ then we merge buffer vertices $\psi_{i.1}, \psi_{i.2}, \ldots, \psi_{i.t}$ into one super node i. This results in

$$d_i = \sum_{h \in [t]} d_{\psi_{i.h}} = \sum_{h \in [t]} y_{i.h} = y_i \leq 1$$

This ensures that at most one copy of any split facility in F_{old} is opened.

Since the fractional degree of the root is $k - |M|$ exactly k facilities are opened. This holds in particular because our gadgets for the split vertices ensures that at most one copy of each split vertex is opened. The remaining above-described gadgets ensure that the properties of the original tree-based rounding used in the analysis by Charikar and Li continue to hold. In particular notice that the negative correlation properties continue to hold because all nodes representing elements $a \in A$ are connected to the root r.

Figure 6 shows the resulting bipartite graph for the instance in Figure A.1.2.

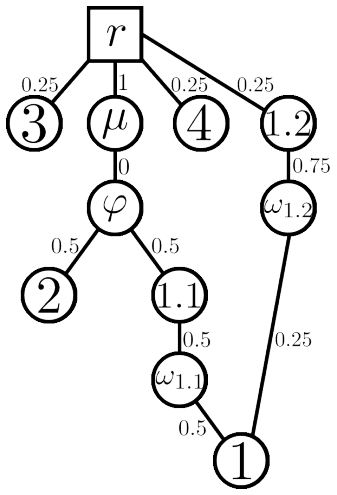


Figure 6: Bipartite graph of the example in A.1.2

A.2 Local Search Approximation Algorithm

In this section we we show that a single-swap local search achieves a constant factor approximation for EXACT-k-FL. To prove a constant factor approximation, we follow the proof of Zhang (2007) and slightly modify it by extending some Lemmas to match the case of Exact-k-UFL. First, we define the following notations. Let $O_{k\text{-FL}} = \{o_1, o_2, \ldots, o_k\}$ be the global optimum and let $S = \{i_1, i_2, \ldots, i_k\}$ be a local optimum. For the sake of notation, we write $O_{k\text{-FL}}$ and $\text{cost}_{k\text{-FL}}$ as simply O and cost respectively when the meaning is clear from the context. Let U be any solution, then the following notations can be introduced:

1. $U_j = d(j, \phi_U(j))$ denotes the service cost for j in U.
2. $N_U(i) = \{j \in C : \phi_U(j) = i\}$, i.e. $N_U(i)$ is the set of clients, which are connected to the facility i in solution U.
3. $N_i^o = N_S(i) \cap N_O(o)$ contains all clients that are served by i and o.
4. We say i captures o, if $|N_i^o| > \frac{1}{2}|N_O(o)|$. In addition, i is called good if it does not capture any o, and bad if it does.
5. We also use a bijective mapping $\pi : N_O(o) \to N_O(o)$. For every $i \in \{i : N_i^o \neq \emptyset\}$ that does not capture o, we have that every $j \in N_i^o$ is mapped outside of N_i^o, i.e., $\pi(j) \notin N_i^o$. If i captures o, then for each $j, \pi(j) \in N_i^o$, we have $\pi(j)$ is mapped back onto j, i.e., $\pi(\pi(j)) = j$. When constructing π as described by Zhang (2007), it also yields the property that if i captures o, then $j = \pi(\pi(j))$ for every $j \in N_O(o)$.

Abusing the notation, for a set of facilities F' and a client j, we let F'_j be the shorthand for $\min_{i \in F'}\{d(i, j)\}$. We first show the following lemma.

Lemma A.1. *Let $j \in N_S(i)$ and $\pi(j) \notin N_S(i)$. After a swap(i, o), the new service cost for the client j can be bounded by $S_{\pi(j)} + O_{\pi(j)} + O_j$.*

Proof. We consider the cases $j \in N_i^o$ and $j \notin N_i^o$ separately:

1. $j \in N_i^o$: Let i' be the nearest facility serving the client $\pi(j)$. After o is swapped in for i, each $j \in N_i^o$ will be served by its new nearest facility $\hat{i}$. By triangle inequality, we have

$$\begin{aligned} d(j, \hat{i}) &\leq d(j, i') \leq d(\pi(j), i') + d(j, \pi(j)) \\ &\leq d(\pi(j), i') + d(\pi(j), o) + d(j, o) \\ &= S_{\pi(j)} + O_{\pi(j)} + O_j \end{aligned}$$

2. $j \notin N_i^o$: Let o' be the facility, with $j \in N_i^{o'}$ and therefore $\pi(j) \in N_i^{o'}$. Furthermore, let i' be the nearest facility serving the client $\pi(j)$. After o is swapped in for i, each $j \notin N_i^o$ will be served by its new nearest facility $\hat{i}$. Again, we can use triangle inequality to obtain the following:

$$\begin{aligned} d(j, \hat{i}) &\leq d(j, i') \leq d(\pi(j), i') + d(j, \pi(j)) \\ &\leq d(\pi(j), i') + d(\pi(j), o') + d(j, o') \\ &= S_{\pi(j)} + O_{\pi(j)} + O_j \end{aligned}$$

□

Lemma A.2. *$S_j \leq S_{\pi(j)} + O_{\pi(j)} + O_j$ for each $j \in C$.*

Proof. Let i' be the nearest facility serving the client $\pi(j)$ and o be the facility for which $j \in N_i^o$. Because π is a mapping from $N_O(o)$ to $N_O(o)$, we know that $\pi(j) \in N_i^o$. Therefore,

$$\begin{aligned} S_j = d(i, j) &\leq d(i', j) \leq d(\pi(j), i') + d(j, \pi(j)) \\ &\leq d(\pi(j), i') + d(\pi(j), o) + d(j, o) \\ &= S_{\pi(j)} + O_{\pi(j)} + O_j \end{aligned}$$

□

Lemma A.3. *If o is the nearest facility that i captures and i also captures $o' \neq o$, then after a swap(i, o), the new service cost for each $j \in N_i^{o'}$ with $\pi(j) \in N_S(i)$ can be bounded by $2S_j + O_j$.*

Proof. Let $\hat{i}$ be the new nearest facility to j. Because of triangle inequality and the fact that o is the nearest facility that i captures, the following holds

$$d(j, \hat{i}) \leq d(j, o) \leq d(j, i) + d(i, o) \leq d(j, i) + d(i, o') \leq d(j, i) + d(j, i) + d(j, o') = 2S_j + O_j.$$

□

For sake of completeness, we also show the following:

Lemma A.4. *After a swap(i, o), the new service cost for a client $j \in N_O(o)$ with $\pi(j) \in N_S(i)$ can be bounded by O_j.*

Proof. Let $\hat{i}$ be the new nearest facility to j. Since $j \in N_O(o)$ is served by o in the solution O, we obtain $d(j, \hat{i}) \leq d(j, o) = O_j$ □

Let cost_s and cost_f denote the *service* and *facility opening* cost (or facility cost, for short) of the given k-FL instance. We next bound the facility cost of the local optimal solution.

Lemma A.5. *The facility cost $\text{cost}_f(S)$ is bounded by $\text{cost}_f(O) + 2\text{cost}_s(O)$.*

Proof. First, partition S into subsets $(A_1, A_2, \ldots, A_m)$ and O into subsets $(B_1, B_2, \ldots, B_m)$ to get pairs (A_i, B_i) with $|A| = |B| \; \forall i \in [m]$. To do this, we pick any bad facility $b \in S$ and add it to A. Afterwards, add every facility $o \in O$ that i captures to B and fill A with arbitrary good facilities in S until $|A| = |B|$. Repeat this for every bad facility in A. Let A_m be the set of good facilities left in S and B_m the set of facilities left in O. Denote with $e \in B$ the facility closest to b. Note that this method of partitioning is sound since no two $i \in S$ can capture the same o and therefore the number of bad facilities in S are $\leq |A|$. We can now apply Lemmas A.4, A.3 and A.1 to bound the cost after swap(b, e) by:

$$\begin{aligned} &- f_b + f_e + \sum_{\substack{j \in N_b^e \\ \pi(j) \in N_S(b)}} (O_j - S_j) + \sum_{\substack{j \in N_S(b) - N_O(e) \\ \pi(j) \in N_S(b)}} (2S_j + O_j - S_j) \\ &+ \sum_{\substack{j \in N_S(b) \\ \pi(j) \notin N_S(b)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0, \end{aligned}$$

which implies,

$$-f_b + f_e + \sum_{\substack{j \in N_b^e \\ \pi(j) \in N_S(b)}} 2O_j + \sum_{\substack{j \in N_S(b) - N_O(e) \\ \pi(j) \in N_S(b)}} (S_j + O_j) + \sum_{\substack{j \in N_S(b) \\ \pi(j) \notin N_S(b)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0$$

Note that this cost is non-negative due to the local optimality of S. Moreover, the cost of a solution after one swap(i, o) with $i \in A - b$ and $o \in B - e$ can be bounded via Lemmas A.4 and A.1 as:

$$-f_i + f_o + \sum_{\substack{j \in N_b^o \\ \pi(j) \in N_S(b)}} (O_j - S_j) + \sum_{\substack{j \in N_i^o \\ \pi(j) \notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0$$

Since $O_j - S_j \leq 2O_j$ and $\{j \in N_b^o : o \in B - e \wedge \pi(j) \in N_S(b)\} = \{j \in N_S(b) - N_O(e) : \pi(j) \in N_S(b)\}$, summing all of the different swaps between A and B gives the following bound:

$$-\sum_{i \in A} f_i + \sum_{o \in B} f_o + \sum_{\substack{j \in N_b^e \\ \pi(j) \in N_S(b)}} 2O_j + \sum_{\substack{j \in N_S(b) - N_O(e) \\ \pi(j) \in N_S(b)}} (S_j + O_j) + \sum_{o \in B - e} \sum_{\substack{j \in N_b^o \\ \pi(j) \in N_S(b)}} (O_j - S_j) + \sum_{i \in A} \sum_{\substack{j \in N_S(b) \\ \pi(j) \notin N_S(b)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0.$$

This implies

$$-\sum_{i \in A} f_i + \sum_{o \in B} f_o + \sum_{\substack{j \in N_S(b) \\ \pi(j) \in N_S(b)}} 2O_j + \sum_{i \in A} \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0.$$

Because every facility $i \in A_m$ is good, we have that for all $i \in A_m$, and for all $j \in N_S(i)$, $\pi(j) \notin N_S(i)$. Therefore, the cost of any swap between A_m and B_m can be bounded using Lemma A.1 by:

$$-\sum_{i \in A} f_i + \sum_{o \in B} f_o + \sum_{i \in A} \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0$$

Finally adding all swaps between A and B results in the following bound:

$$
\begin{aligned}
& -\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{t=1}^{m} \sum_{\substack{j\in N_S(b_t)\\ \pi(j)\in N_S(b_t)}} O_j \\
& + \sum_{i\in A} \sum_{\substack{j\in N_S(i)\\ \pi(j)\notin N_S(i)}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) \geq 0 \\
\implies & -\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{t=1}^{m} \sum_{\substack{j\in N_S(b_t)\\ \pi(j)\in N_S(b_t)}} O_j \\
& + \sum_{\substack{j\in C\\ \pi(j)\notin N_S(\phi_S(i))}} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j)) \geq 0 \\
\implies & -\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{t=1}^{m} \sum_{\substack{j\in N_S(b_t)\\ \pi(j)\in N_S(b_t)}} O_j + 2 \sum_{\substack{j\in C\\ \pi(j)\notin N_S(\phi_S(i))}} O_j \geq 0 \\
\implies & -\sum_{i\in A} f_i + \sum_{o\in B} f_o + 2\sum_{j\in C} O_j \geq 0 \\
\implies & -cost_f(S) + cost_f(O) + 2cost_s(O) \geq 0 \\
\implies & cost_f(S) \leq cost_f(O) + 2cost_s(O).
\end{aligned}
$$

□

Lemma A.6. *If only one swap at a time is allowed, the service cost for the local optimum S can be bounded by* $cost_s(S) \leq cost_f(O) + 5cost_s(O)$.

Proof. First, partition S into subsets $(A_1, A_2, \ldots, A_m)$ and O into subsets $(B_1, B_2, \ldots, B_m)$ with the same method presented in Lemma A.5. Now, we again bound the cost after a swap(i, o) with the key difference that $i \in A - b$, $o \in B$. To obtain a term that sums up all S_j's for $j \in C$, which is the service cost $\text{cost}_s(S)$, we bound the cost after a swap(i, o) slightly looser than in Lemma A.5. Since no $i \in A - b$ captures an $o \in O$, we know that $\pi(j) \notin N_S(i)$ for each $j \in N_S(i)$. Therefore, we can bound a swap(i, o) using Lemmas A.1 and A.2 by:

$$
\begin{aligned}
& -f_i + f_o + \sum_{j\in N_O(o)} (O_j - S_j) + \sum_{j\in N_S(i)-N_O(o)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) && \geq 0 \\
\implies & -f_i + f_o + \sum_{j\in N_O(o)} (O_j - S_j) + \sum_{j\in N_S(i)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) && \geq 0.
\end{aligned}
$$

Summing up swap(i, o) for $i \in A - b, o \in B - e$ and one swap(i', e) for some $i' \in A - b$ and again using Lemma A.2 results in

$$
\begin{aligned}
& -\sum_{i\in A-b} f_i + \sum_{o\in B} f_o + \sum_{o\in B} \sum_{j\in N_O(o)} (O_j - S_j) \\
& + \sum_{i\in A-b} \sum_{j\in N_S(i)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) + \sum_{j\in N_S(i')} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) && \geq 0 \\
\implies & -\sum_{i\in A-b} f_i + \sum_{o\in B} f_o + \sum_{o\in B} \sum_{j\in N_O(o)} (O_j - S_j) \\
& + 2\sum_{i\in A} \sum_{j\in N_S(b)} (S_{\pi(j)} + O_{\pi(j)} + O_j - S_j) && \geq 0 \\
\implies & \sum_{o\in B} f_o + 5\sum_{j\in C} O_j - \sum_{j\in C} S_j && \geq 0.
\end{aligned}
$$

By rearranging the terms we get

$$\sum_{j \in C} S_j \leq \sum_{o \in B} f_o + 5 \sum_{j \in C} O_j.$$

implying the claim of the lemma $cost_s(S) \leq cost_f(O) + 5cost_s(O)$. □

Theorem A.7. *A local search algorithm for Exact-k-UFL with a single swap operation, has a locality gap of at most 7.*

Proof.

$$\begin{aligned} cost(S) &= cost_f(S) + cost_s(S) \\ &\leq cost_f(O) + 2cost_s(O) + cost_f(O) + 5cost_s(O) \\ &\leq 7(cost_f(O) + cost_s(O)) \\ &= 7cost(O). \end{aligned}$$

□

B OMITTED PROOFS OF SECTION 3

We delay the proofs of Lemma 3.4 and Theorem 3.1 to the end of this section as they heavily depend on the technical lemmas that proceed them in the main body. Once we prove these lemmas, we can move on to present how to derive the proof of Lemma 3.4 from these technical lemmas and standard techniques in the literature. Finally, Theorem 3.1 can be shown directly as a consequence of Lemmas 2.4, 2.5, and 3.4.

Proof of Lemma 3.5. For a client $j \in C$, let $\sigma(j)$ and $\sigma^*(j)$ denote the closest facility to j in S_t and $O_{k\text{-FL}}$, respectively. First, we bound $\sum_{j \in C} d(j, S_t)$. Moreover, let j' denote the client in T_t that is closest to j and let i' be the facility that j' places in S_t.

$$\begin{aligned} d(j, S_t) = d(j, \sigma(j)) &\leq d(j, i') \\ &\leq d(j, j') + d(j', i') && (\Leftarrow \text{by triangle inequality}) \\ &\leq d(j, j') + d(j', \sigma^*(j)) && (\Leftarrow \text{by the way } i^* \text{ is added to } S_t) \\ &\leq d(j, j') + d(j, j') + d(j, \sigma^*(j)) && (\Leftarrow \text{by triangle inequality}) \end{aligned}$$

Summing over all clients,

$$\begin{aligned} \sum_{j \in C} d(j, S_t) &\leq \sum_{j \in C} \left(2d(j, T_t) + d(j, O_{k\text{-FL}})\right) \\ &\leq (4\Delta + 1) \sum_{j \in C} d(j, O_{k\text{-FL}}). \end{aligned} \tag{1}$$

Next, by triangle inequality, we have that

$$\begin{aligned} \sum_{j \in C} d(j, S_t) &= \sum_{j \in C} d(j, \sigma(j)) \\ &\geq \sum_{j \in C} \left(d(\sigma(j), \sigma^*(j)) - d(j, \sigma^*(j))\right). \end{aligned} \tag{2}$$

Combining Inequalities 1 and 2, we get:

$$\sum_{j \in C} d(\sigma(j), \sigma^*(j)) \leq (4\Delta + 2) \sum_{j \in C} d(j, \sigma^*(j)). \tag{3}$$

We now construct a set $S' \subseteq S_t$ of cardinality at most k with a good connection cost. For every facility $i \in O_{k\text{-FL}}$, let $s(i)$ be the facility in $S_t \cup \{i^*\}$ that is closest to i. Define $S' := \{s(i) : i \in O_{k\text{-FL}}\}$. Using the triangle inequality one more time, we get

$$\begin{aligned}\sum_{j\in C} d(j, s(\sigma(j))) &\le \sum_{j\in C} \big(d(j,\sigma(j)) + d(\sigma(j),\sigma^*(j)) + d(\sigma^*(j), s(\sigma^*(j)))\big)\\ &\le \sum_{j\in C} \big(d(j,\sigma(j)) + 2d(\sigma(j),\sigma^*(j))\big)\\ &\le (12\Delta+5)\sum_{j\in C} d(j,\sigma^*(j)),\end{aligned} \tag{4}$$

where in the second inequality, we used the definition of $s(\sigma^*(j))$, and in the last inequality, we used the bound in Inequalities 1 and 3. As a result, S' has the desired connection cost. It remains to show it also has a relatively small disagreement cost. To see this, take any facility $i \in S'$, and let $s^{-1}(i) \in O_{k\text{-FL}}$ be the facility in the optimum set that was responsible for adding i to S', breaking ties arbitrarily. Note that $d(s^{-1}(i), i) \le d(s^{-1}(i), i^*)$ because the anchor i^* is a candidate for the minimizer in the definition of $s(i)$. So, for the opening cost of i, we can use triangle inequality and get

$$\begin{aligned}d(i,i^*) &\le d(i, s^{-1}(i)) + d(s^{-1}(i), i^*)\\ &\le 2d(s^{-1}(i), i^*).\end{aligned}$$

Summing over all $i \in S'$ and noting that every facility in $O_{k\text{-FL}}$ is accounted for in the sum at most once, we get

$$\lambda k \sum_{i\in S'} d(i,i^*) \le 2\lambda k \sum_{i\in O_{k\text{-FL}}} d(i,i^*). \tag{5}$$

Inequalities 4 and 5 together complete the proof of the lemma. □

Proof of Lemma 3.6.

$$\begin{aligned}\mathbb{P}\left[j \text{ is sampled from } C_\ell \text{ where } C_\ell \in \mathsf{Bad}_t\right] &= \frac{\sum_{C_\ell\in\mathsf{Bad}_t}\sum_{j\in C_\ell} d(j,T_{t-1})}{\sum_{j\in C} d(j,T_{t-1})}\\ &= 1 - \frac{\sum_{C_\ell\in\mathsf{Good}_t}\sum_{j\in C_\ell} d(j,T_{t-1})}{\sum_{j\in C} d(j,T_{t-1})}\\ &\ge 1 - \frac{\sum_{C_\ell\in\mathsf{Good}_t}\sum_{j\in C_\ell} d(j,T_{t-1}\cup\{i^*\})}{\sum_{j\in C} d(j,T_{t-1})}\\ &\ge 1 - \frac{\Delta\cdot\sum_{C_\ell\in\mathsf{Good}_t}\sum_{j\in C_\ell} d(j,O_{k\text{-FL}})\}}{2\Delta\cdot\sum_{j\in C} d(j,O_{k\text{-FL}})}\\ &\ge 1/2.\end{aligned}$$

□

Proof of Observation 1. Assume $j_\ell \in \text{core}(C_\ell)$ is the sampled client. Then

$$\begin{aligned}\sum_{j\in C_\ell} d(j,i'_\ell) &\le \sum_{j\in C_\ell} d(j,j_\ell) + d(j_\ell,i'_\ell)\\ &\le \sum_{j\in C_\ell} \big(d(j,j_\ell) + d(j_\ell,i_\ell)\big)\\ &\le \sum_{j\in C_\ell} \big(d(j,i_\ell) + 2d(j_\ell,i_\ell)\big)\\ &\le \sum_{j\in C_\ell} \big(d(j,i_\ell) + 2\alpha\cdot r_\ell\big)\\ &\le (2\alpha+1)\sum_{j\in C_\ell} d(j,i_\ell) = (2\alpha+1)\sum_{j\in C_\ell} d(j,\sigma^*(j)),\end{aligned}$$

where in the second inequality, we used the fact that i'_ℓ is closer to j_ℓ than i_ℓ by definition, in the third inequality we used the metric property $d(j, j_\ell) \leq d(j, i_\ell) + d(j_\ell, i_\ell)$, in the fourth inequality we used the fact that $j_\ell \in \text{core}(C_\ell)$, and in the fifth inequality we used the definition of r_ℓ. □

Proof of Observation 2. Since $C_\ell \in \mathsf{Bad}_t$, we have

$$\begin{aligned}
\Delta \cdot |C_\ell| \cdot r_\ell &= \Delta \cdot \sum_{j \in C_\ell} d(j, \sigma^*(j)) \\
&< \sum_{j \in C_\ell} d(j, T_{t-1}) \\
&\leq \sum_{j \in C_\ell} \big(d(j, i_\ell) + d(i_\ell, T_{t-1})\big) \\
&= \sum_{j \in C_\ell} \big(d(j, i_\ell) + \delta_{\ell,t}\big) \\
&= |C_\ell| \cdot r_\ell + |C_\ell| \cdot \delta_{\ell,t}.
\end{aligned}$$

We get the claim of the observation by rearranging the terms. □

Proof of Lemma 3.8.

$$\begin{aligned}
\sum_{j \in C_\ell} d(j, \sigma^*(j)) &\geq \sum_{j \in C_\ell \setminus \text{core}(C_\ell)} d(j, \sigma^*(j)) \\
&> \sum_{j \in C_\ell \setminus \text{core}(C_\ell)} \alpha \cdot r_\ell \\
&= |C_\ell \setminus \text{core}(C_\ell)| \cdot \alpha \cdot r_\ell \\
&= (|C_\ell| - |\text{core}(C_\ell)|) \cdot \alpha \cdot \frac{\sum_{j \in C_\ell} d(j, \sigma^*(j))}{|C_\ell|}.
\end{aligned}$$

Rearranging the terms will give us the claimed statement. □

Proof of Lemma 3.9. Assume we are sampling a client from a cluster $C_\ell \in \mathsf{Bad}_t$. The probability of this client being in $\text{core}(C_\ell)$ is $\left(\sum_{j \in \text{core}(C_\ell)} d(j, T_{t-1})\right) / \left(\sum_{j \in C_\ell} d(j, T_{t-1})\right)$. We prov the lemma by lower-bounding the numerator and upper-bounding the denominator.Using the triangle inequality and assuming that $s(j)$ denotes the closest client in T_{t-1} to j we have

$$\begin{aligned}
\sum_{j \in \text{core}(C_\ell)} d(j, T_{t-1}) &= \sum_{j \in \text{core}(C_\ell)} d(j, s(j)) \\
&\geq \sum_{j \in \text{core}(C_\ell)} \big(d(s(j), \sigma^*(j)) - d(j, \sigma^*(j))\big) \\
&\geq \sum_{j \in \text{core}(C_\ell)} \big(\delta_{\ell,t} - \alpha \cdot r_\ell\big) \\
&= |\text{core}(C_\ell)| \cdot \big(\delta_{\ell,t} - \alpha \cdot r_\ell\big) \\
&= \left(1 - \frac{1}{\alpha}\right) |C_\ell| \left(\delta_{\ell,t} - \alpha \cdot r_\ell\right), \qquad (6)
\end{aligned}$$

where in the first inequality, we used the definition of $\delta_{\ell,t}$, and in the last equality, we applied Lemma 3.8 to $|\text{core}(C_\ell)|$. For the denominator, we have

$$\begin{aligned}
\sum_{j \in C_\ell} d(j, T_{t-1}) &\leq \sum_{j \in C_\ell} \big(d(j, \sigma^*(j)) + d(\sigma^*(j), T_{t-1})\big) \\
&= |C_\ell| \left(r_\ell + \delta_{\ell,t}\right). \qquad (7)
\end{aligned}$$

Dividing (6) by (7), we get

$$\mathbb{P}\left[\text{sample } j \in \text{core}(C_\ell) \mid j \in C_\ell \text{ and } C_\ell \in \mathsf{Bad}_t\right] = \frac{\sum_{j \in \text{core}(C_\ell)} d(j, T_{t-1})}{\sum_{j \in C_\ell} d(j, T_{t-1})} \geq \frac{\left(1 - \frac{1}{\alpha}\right) |C_\ell| \left(\delta_{\ell,t} - \alpha \cdot r_\ell\right)}{|C_\ell| \left(r_\ell + \delta_{\ell,t}\right)}. \tag{8}$$

From Observation 2 and the choice of α, we know that $\delta_{\ell,t} \geq (\Delta - 1) \cdot r_\ell \geq \alpha \cdot r_\ell$. As a result, the right-hand-side of Inequality 8 is an increasing function in $\delta_{\ell,t}$, and therefore we can lower-bound it by setting $\delta_{\ell,t}$ to its minimum possible value, namely, $(\Delta - 1) \cdot r_\ell$:

$$\mathbb{P}\left[j \in \text{core}(C_\ell) \mid j \in C_\ell \text{ and } C_\ell \in \mathsf{Bad}_t\right] \geq \left(1 - \frac{1}{\alpha}\right) \frac{\Delta - 1 - \alpha}{\Delta}.$$

□

Proof of Observation 3. Let j_ℓ denote the facility that we have sampled from near_core(C_ℓ) for a Bad cluster C_ℓ. Then

$$\begin{aligned}
d(i', i^*) &\leq d(i', j_\ell) + d(j_\ell, i^*) && (\Leftarrow \text{by triangle inequality}) \\
&\leq d(i', j_\ell) + \gamma \cdot d(i_\ell, i^*) && (\Leftarrow \text{since } j \in \text{near_core}(C_\ell)) \\
&\leq d(i_\ell, j_\ell) + \gamma \cdot d(i_\ell, i^*) && (\Leftarrow \text{by the choice of } i') \\
&\leq d(i_\ell, i^*) + d(i^*, j_\ell) + \gamma \cdot d(i_\ell, i^*) && (\Leftarrow \text{by triangle inequality}) \\
&\leq (2\gamma + 1) \cdot d(i_\ell, i^*). && (\Leftarrow \text{by using the fact that } j \in \text{near_core}(C_\ell) \text{ again})
\end{aligned}$$

□

Proof of Lemma 3.12. To show this lemma, we provide a lower bound as well as an upper bound on the connection cost of the clients in the far_core(C_ℓ) in an optimum solution. First, note that by the definition of the far core we have

$$\sum_{j \in \text{far\_core}(C_\ell)} d(j, i_\ell) \leq \sum_{j \in \text{far\_core}(C_\ell)} \alpha \cdot r_\ell \leq |\text{far\_core}(C_\ell)| \, (\alpha \cdot r_\ell) \leq \alpha \cdot |\text{core}(C_\ell)| \cdot r_\ell. \tag{9}$$

Next, by using the triangle inequality and definition of far core, we can write

$$\begin{aligned}
\sum_{j \in \text{far_core}(C_\ell)} d(j, i_\ell) &\geq \sum_{j \in \text{far_core}(C_\ell)} \left(d(j, i^*) - d(i_\ell, i^*)\right) \\
&\geq \sum_{j \in \text{far_core}(C_\ell)} \left(\gamma \cdot d(i_\ell, i^*) - d(i_\ell, i^*)\right) \\
&\geq |\text{far_core}(C_\ell)| \cdot (\gamma - 1) \cdot d(i_\ell, i^*) \\
&\geq (1 - \rho) \cdot |\text{core}(C_\ell)| \cdot (\gamma - 1) \cdot d(i_\ell, i^*).
\end{aligned} \tag{10}$$

Putting Inequalities 9 and 10 together, we arrive at:

$$d(i_\ell, i^*) \leq \frac{\alpha}{(1 - \rho) \cdot (\gamma - 1)} \cdot r_\ell \leq \alpha \cdot r_\ell,$$

when we set the parameter γ and ρ in a way that $(1 - \rho) \cdot (\gamma - 1) \geq 1$. □

Proof of Lemma 3.13. The proof follows from Corollary 3.11, Lemmas 3.6 and 3.12, and the fact that that $\mathbb{P}[A] \geq \mathbb{P}[A \mid B] \times \mathbb{P}[B]$. □

Before presenting the proof of the main result of the sparsification procedure, we first prove our padding lemma, which roughly says that in case our sampling procedure fails to pick at least k facilities in the end, we can pad it with some facilities in the ball near the anchor i^* and compensate for the cardinality without increasing the cost too mush.

Proof of Lemma 3.14. Let $S' = S \cup A$ be the padded sample set, where $A = \{\max\{0, k-|S|\}$ facilities from$F$ to $i^*\}$. We propose an assignment of the clients to S' whose cost satisfies the statement of the lemma. Obviously, the best assignment of clients to facilities could only have identical or smaller cost. This would imply the claim of the lemma. The trick is that we do not assign any clients to the auxiliary facilities in A. As a result, $\sum_{j \in C} d(j, S') = \sum_{j \in C} d(j, S)$, meaning that we have not increased the connection cost. Regarding the disagreement cost, $\sum_{i \in A} d(i, i^*) \leq \sum_{i \in O_{k\text{-FL}}} d(i, i^*)$ since the facilities in A are the closest ones to i^*. Adding the reconciliation cost coming from the facilities in S will complete the proof. □

We are now ready to prove Lemma 3.4, and consequently, our main result about the sampling procedure, Theorem 3.1.

Proof of Lemma 3.4. Let $p := \frac{\rho}{2} \cdot \left(1 - \frac{1}{\alpha}\right)\left(1 - \frac{\alpha+1}{\Delta}\right) \approx 0.393$. Following the steps of the proof of Theorem 1 by Aggarwal et al. (2009), we can show that of we keep sampling clients for μk rounds for a constant value of μ satisfying $\mu k \geq (k + \sqrt{k})/p$, we will have that $\mathbb{P}\left[|\mathsf{Bad}_{\mu k}| \geq 0\right] \leq 1 - \exp(-p/4)$. □

Proof of Theorem 3.1. Let S_{i^*} denote the result of the sampling for an iteration of the second phase where $i = i^*$. By Lemma 3.4, with constant probability there exists $S' \subseteq S$ with $|S'| = k$ and $\text{cost}_{k\text{-FL}}(S') = \mathcal{O}(\text{OPT}_{k\text{-FL}})$, so S' is a constant-factor approximation for k-FL. Conditioned on this event, we then get $\text{cost}_{\text{RC}}(S') = \mathcal{O}(\text{OPT}_{k\text{-FL}})$ by Lemma 2.4. Next, by applying Lemma 2.5, we have that $\text{cost}_{\text{RC}}(S') = \mathcal{O}(\text{OPT}_{\text{RC}})$ with constant probability. Since such a solution exists among the facilities of S_{i^*}, ALG_{RC} must be able to find a solution S'' with $\text{cost}_{\text{RC}}(S'') = \mathcal{O}(\text{cost}_{\text{RC}}(S')) = \mathcal{O}(\text{OPT}_{\text{RC}})$. Finally, the constant probability can be boosted to any arbitrary large probabilities by repeating the algorithm polynomially many times and returning the best output at the end. □

C UNBOUNDED LOCALITY GAP FOR THE CASE $C \neq F$

In the following we will show that a multi-swap local search algorithm cannot provide a bound on locality gap. Multi-swap local search works considers p-swaps at each iteration, Here, p is constant indicating the number of facilities that can be swapped at a single iteration.

Theorem C.1. *There exist instances of* REC k-MEDIAN *problem for which there is a locally optimal solution S with respect to p-swaps for any constant p with* $\text{cost}_{\text{RC}}(S) \geq z \cdot \text{OPT}_{\text{RC}}$ *for an arbitrarily large z.*

Proof. Let $F = A \cup B$, $A = \{a_1, a_2, \ldots, a_k\}$, $B = \{b_1, b_2, \ldots, b_k\}$, $C = \{1, 2\}$, $k > p$ and $z \in \mathbb{R}$. Let the distance function d be as follows:

1. $\forall i, j, h \in [k], i \neq h : d(a_i, a_i) = d(b_i, b_i) = 0$, $d(a_i, a_h) = d(b_i, b_h) = 1$ and $d(a_i, b_j) = z$
2. $\forall i \in [k] \wedge j \in C : d(a_i, j) = 1$ and $d(b_i, j) = z$

Again, having distance 1 between facilities in the same set (A or B) and the distance z between facilities in different sets. Clients and facilities in A have a distance of 1 and Clients and facilities in B have a distance of z (see 7). Then, the global optimum $O = A$ and the goal is for B to be a local optimum S.

$$\text{cost}_{\text{RC}}(O) = \lambda\frac{k^2 - k}{2} + 2 < \lambda\frac{k^2 - k}{2} + 2z = \text{cost}_{\text{RC}}(S)$$

Note that $\text{cost}_{\text{RC}}(S)/\text{cost}_{\text{RC}}(O)$ tends to z for large enough values of z. Let $A_i \subseteq A$ and $B_i \subseteq B$, with $|A_i| = |B_i| = i$. Due to the symmetry of the facilities in A and in B, it does not matter which $a \in A$ is included in A_i and which $b \in B$ is included in B_i. For $q \in \mathbb{N}$ and $q \leq p$, we consider the neighbourhood of S, consisting of the neighbours that are one swap(A_q, B_q) away. Due to the symmetry between a's and the symmetry between b's, all neighbours obtained by a swap(A_q, B_q) have the same cost. Since $S + A_q - B_q$ has exactly q many a's and $k - q$ many b's, the edges are as follows:

- $\frac{(k^2-k)}{2}$ many edges in total.
- $q(k - q) = qk - q^2$ many edges between $a \in A_q$ and $b \in S - B_q$.
- $\frac{(k^2-k-2qk+2q^2)}{2}$ many edges between facilities in the same set.

Thus, we obtain:

$$\text{cost}_{\text{RC}}(S + A_q - B_q) = \lambda \left(z(qk - q^2) + \frac{k^2 - k - 2qk + 2q^2}{2} \right) + 2$$

To ensure S being a local optimum, the following has to hold for all $q \in [p]$:

$$\begin{aligned} cost(S + A_q - B_q) &> cost(S) \\ \lambda z(qk - q^2) + \lambda \frac{k^2 - k - 2qk + 2q^2}{2} + 2 &> \lambda \frac{k^2 - k}{2} + 2z \\ z(\lambda(qk - q^2) - 2) &> \lambda(\frac{2qk - 2q^2}{2}) - 2 \\ z(\lambda(qk - q^2) - 2) &> \lambda(qk - q^2) - 2 \\ z &> 1, \text{ for } \lambda > \frac{2}{qk - q^2} \end{aligned}$$

The function $f_k(q) = qk - q^2$ is a parabola, with $f_k(q) = 0$ for $q = 0$ and $q = k$, and $f_k(q) > 0$ for $0 < q < k$. The parabola's only turning point, which is a maximum, is at $k/2$, therefore maximizing $2/(qk - q^2)$ for $q = 1$. Hence, λ has to be greater than $\frac{2}{k-1}$. However, as $k \to \infty$, $\lambda \to 0$, enabling the gap between S and O to approach infinity, even for small λ's. □

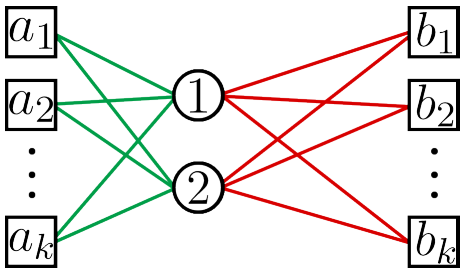


Figure 7: This figure depicts, the distances between the clients C and the facilities F. A red line indicates a distance of z and a green line indicates a distance of 1

D AN AGNOSTIC APPROXIMATION FOR REC k-MEDIAN

We consider the case $C = F$.

Consider an arbitrary solution $S = \{s_1, \ldots, s_k\}$ and an optimal solution $O = \{o_1, \ldots, o_k\}$. We define $c(s) \in O$ to be the facility serving s in the optimal solution, or s if $s \in O$. We denote the service and reconciliation cost of solution S as $f(S)$ and $g(S)$ respectively, the total cost being $f(S) + \frac{\lambda}{2} g(S)$. We assume λ to be constant.

We consider the reconciliation cost $g(S) = \sum_i \sum_j d(s_i, s_j)$ and bound each term as follows:

$$d(s_i, s_j) \leq d(s_i, c(s_i)) + d(c(s_i), o_i) + d(o_i, o_j) + d(o_j, c(s_j)) + d(c(s_j), s_j).$$

We thus have

$$\begin{aligned} g(S) &\leq \sum_i \sum_j d(s_i, c(s_i)) + d(c(s_i), o_i) + d(o_i, o_j) + d(o_j, c(s_j)) + d(c(s_j), s_j) \\ &= \sum_i \sum_j d(s_i, c(s_i)) + d(c(s_i), o_i) + \sum_i \sum_j d(o_i, o_j) + \sum_i \sum_j d(o_j, c(s_j)) + d(c(s_j), s_j) \\ &= k \left(\sum_i d(s_i, c(s_i)) + d(c(s_i), o_i) \right) + \sum_i \sum_j d(o_i, o_j) + k \left(\sum_j d(o_j, c(s_j)) + d(c(s_j), s_j) \right) \\ &\leq k \left(f(O) + g(O) \right) + g(O) + k \left(f(O) + g(O) \right) \\ &= 2k f(O) + (2k + 1) g(O). \end{aligned}$$

Figure 8: Effects of the sparsification method on the WIKIELECTIONS dataset. On the left, we report the cost yielded by k-UFL+LS, for different amounts of sampled clients. In this experiment, the costs achieved by LOCALSEARCH were similar. On the right, we report the running times.

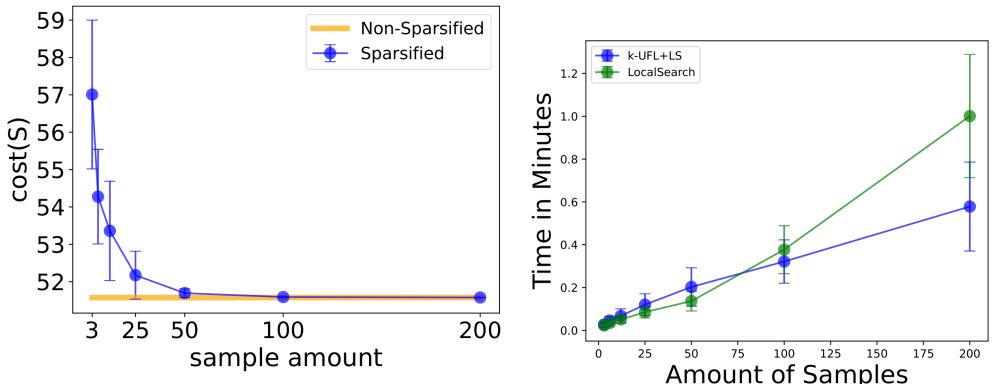


Table 3: Characteristics of the TWITTER dataset.

Name	Number of clients	Number of facilities
TWITTER	3 302 362	500

Assume now that S was found by using an α-approximation algorithm for $f(S)$, where $\alpha = O(k)$. The cost achieved by S is

$$\begin{aligned} f(S) + \frac{\lambda}{2} g(S) &\leq \alpha f(O) + \frac{\lambda}{2}(2kf(O) + (2k+1)g(O)) \\ &\leq \alpha f(O) + \lambda(kf(O) + (k+1/2)g(O)) \\ &= (\alpha + \lambda k) f(O) + \lambda(k+1/2)g(O) \\ &= O(k)OPT. \end{aligned}$$

Thus, a reconciliation-agnostic $O(k)$-approximation algorithm for k-median achieves an $O(k)$-approximation of REC k-MEDIAN.

E EXPERIMENTAL SETUP

We run our experiments on a machine equipped with an Intel(R) Xeon(R) Gold 6248 CPU at 2.50GHz. The CPU has 80 cores, although we limit all our executions to 16 parallel threads. The machine has 784GB of main memory.

F ADDITIONAL RESULTS ON THE EFFECTS OF SPARSIFICATION

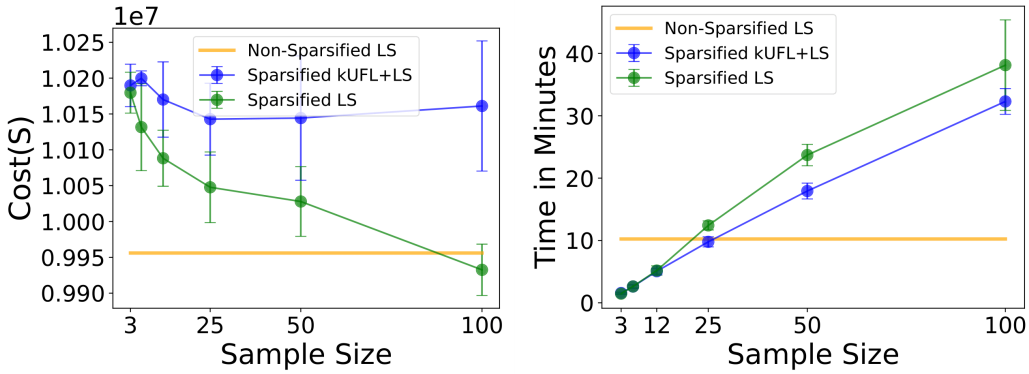
In Figure 8 we plot the results of the sparsification method on the WIKIELECTIONS dataset. We report the costs and running times, as we sample increasing numbers of clients. It can be seen that a few sampled clients suffice to bring the cost down significantly, to a level close to the one obtained when employing the entire dataset.

G ADDITIONAL EXPERIMENTS

In order to further test the benefits of our sparsification approach, we perform additional experiments on a large dataset. In particular, we employ the TWITTER dataset made available by Ordozgoiti and Gionis (2019). It consists of graph data representing a snapshot of the follower network of Twitter. For details, see (Ordozgoiti and Gionis, 2019).

As before, We fix $k = 8$, and vary the value of μ (see Algorithm 1) so that the number of sampled clients is in $\{3, 6, 12, 25, 50, 100\}$. This time, we set $\lambda = \frac{2n}{k(k-1)}$, where n is the number of clients, so that both terms in the objective function are comparable in magnitude. We plot the results in Figure 9. As shown in the plot, the cost for LOCALSEARCH

Figure 9: Effects of the sparsification method on the TWITTER dataset.



drops sharply as the number of sampled clients increases. Interestingly, k-UFL+LS fails to attain results of the same quality. While we are unsure of the reason, we speculate that this might be a consequence of the fact that this algorithm optimizes a proxy of the second term of the objective. While this strategy enables the derivation of a constant-factor approximation guarantee, in practical settings where this term is large it might introduce significant distortions. Thus, in this scenario, we advocate for the choice of the LOCALSEARCH heuristic as a cautious rule of thumb.

The running times, shown as well in Figure 9, illustrate how the sparsification method offers an effective way to trade-off between computational costs and solution quality. We emphasize that in these experiments we limited the processing power to 16 cores. Nevertheless, since the method involves running an instance of the problem for each facility in the dataset, it stands to reason that additional parallel processors would yield significant benefits. Even though we did not have time to systematically test this claim before submission, we did execute several runs, using 25, 50 and 100 sampled clients, using 64 cores. We observed running times of around 200, 400 and 750 seconds respectively, which are comparable or superior to those taken by running the vanilla LOCALSEARCH algorithm on the entire dataset. This reinforces the conclusion that our approach makes it possible to improve running times with moderate sacrifices in quality.