Near-Optimal Dimension Reduction for Facility Location*

Lingxiao Huang†
The State Key Laboratory of Novel Software Technology,
New Cornerstone Science Laboratory
Nanjing University
Nanjing, China
huanglingxiao@nju.edu.cn

Shaofeng H.-C. Jiang‡
Peking University
Beijing, China
shaofeng.jiang@pku.edu.cn

Robert Krauthgamer§
Weizmann Institute of Science
Rehovot, Israel
robert.krauthgamer@weizmann.ac.il

Di Yue
Peking University
Beijing, China
di_yue@stu.pku.edu.cn

Abstract

Oblivious dimension reduction, à la the Johnson-Lindenstrauss (JL) Lemma, is a fundamental approach for processing high-dimensional data. We study this approach for Uniform Facility Location (UFL) on a Euclidean input $X \subset \mathbb{R}^d$, where facilities can lie in the ambient space (not restricted to X). Our main result is that target dimension $m = \tilde{O}(\varepsilon^{-2}\mathrm{ddim})$ suffices to $(1+\varepsilon)$-approximate the optimal value of UFL on inputs whose doubling dimension is bounded by ddim. It significantly improves over previous results, that could only achieve $O(1)$-approximation [Narayanan, Silwal, Indyk, and Zamir, ICML 2021] or dimension $m = O(\varepsilon^{-2}\log n)$ for $n = |X|$, which follows from [Makarychev, Makarychev, and Razenshteyn, STOC 2019].

Our oblivious dimension reduction has immediate implications to streaming and offline algorithms, by employing known algorithms for low dimension. In dynamic geometric streams, it implies a $(1+\varepsilon)$-approximation algorithm that uses $O(\varepsilon^{-1}\log n)^{\tilde{O}(\mathrm{ddim}/\varepsilon^2)}$ bits of space, which is the first streaming algorithm for UFL to utilize the doubling dimension. In the offline setting, it implies a $(1+\varepsilon)$-approximation algorithm, which we further refine to run in time $((1/\varepsilon)^{\tilde{O}(\mathrm{ddim})}d + 2^{(1/\varepsilon)^{\tilde{O}(\mathrm{ddim})}}) \cdot \tilde{O}(n)$. Prior work has a similar running time but requires some restriction on the facilities [Cohen-Addad, Feldmann and Saulpic, JACM 2021].

Our main technical contribution is a fast procedure to decompose an input X into several k-median instances for small k. This decomposition is inspired by, but has several significant differences from [Czumaj, Lammersen, Monemizadeh and Sohler, SODA 2013], and is key to both our dimension reduction and our PTAS.

CCS Concepts

• **Theory of computation → Random projections and metric embeddings**.

Keywords

dimension reduction, facility location, approximation algorithms, streaming and sketching

ACM Reference Format:
Lingxiao Huang, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Di Yue. 2025. Near-Optimal Dimension Reduction for Facility Location. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC '25), June 23–27, 2025, Prague, Czechia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3717823.3718214

1 Introduction

A fundamental approach for dealing with high-dimensional data is *oblivious dimension reduction*, in which the dataset $X \subset \mathbb{R}^d$ is mapped to low dimension using a map chosen independently of the data. A cornerstone of this approach is the Johnson-Lindenstrauss (JL) Lemma [30], which states that for all $n \geq 1$ and $0 < \varepsilon < 1$ there is a randomly chosen linear transformation $\pi : \mathbb{R}^d \to \mathbb{R}^m$ for $m = O(\varepsilon^{-2}\log n)$, such that for every dataset $X \subset \mathbb{R}^d$, $|X| = n$, with high probability all the pairwise distances in X are preserved within $(1 \pm \varepsilon)$-factor, i.e.,

$$\forall x, y \in X, \qquad \|\pi(x) - \pi(y)\|_2 \in (1 \pm \varepsilon)\|x - y\|_2. \tag{1}$$

This bound on the *target dimension* $m = m(\varepsilon, n)$ is known to be asymptotically tight [36]. In algorithmic applications, one typically applies on the input $X \subset \mathbb{R}^d$ a map π that is chosen independently of X, and then executes on $\pi(X) \subset \mathbb{R}^m$ some known algorithm for low dimension. This approach has generally proved to be extremely useful.

However, in several fundamental algorithmic applications, target dimension of the form $m = O(\log n)$ is too high to be effective. We can illustrate this by examples from 3 different computational settings: In offline approximation algorithms, the traveling salesman problem (TSP) in dimension $m = O(\log n)$ does not admit a PTAS (i.e., for a sufficiently small but fixed $\varepsilon_0 > 0$, no polynomial-time

*Full version of the paper is available at arXiv:2411.05432.
†Research partially supported by New Cornerstone Science Foundation.
‡Research partially supported by a national key R&D program of China No. 2021YFA1000900 and a startup fund from Peking University.
§Work partially supported by the Israel Science Foundation grant #1336/23, by the Israeli Council for Higher Education (CHE) via the Weizmann Data Science Research Center, and by a research grant from the Estate of Harry Schutzman.

STOC '25, Prague, Czechia

ACM ISBN 979-8-4007-1510-5/25/06
https://doi.org/10.1145/3717823.3718214

algorithm can achieve $(1+\varepsilon_0)$-approximation), assuming P $\neq$ NP [44]. In streaming algorithms, approximating the value of Euclidean minimum spanning tree (MST) in dimension $m = O(\log n)$ within $(1+\varepsilon_0)$ factor (again, for some fixed $\varepsilon_0 > 0$) requires $\Omega(\sqrt{n})$ bits of storage [11]. In fine-grained complexity, the diameter of a point set in dimension $m = O(\log n)$ cannot be $(1+\varepsilon_0)$-approximated (again, for some fixed $\varepsilon_0 > 0$) in quadratic-time, under some complexity assumption [45].

To break below this barrier of target dimension $m = O(\log n)$, one often seeks better bounds for *specific* computational problems. A prime example is that for k-median and k-means clustering, the dimension can be reduced to $m = \tilde{O}(\varepsilon^{-2}\log k)$ [37].[1] This highly nontrivial bound is significantly stronger than earlier/other bounds [4, 6, 14], and offers a substantial improvement for small k. It has become famous due to its many applications, from faster algorithms through better approximation to coreset constructions, and is useful also in many variants of the problem, like fair clustering. Other problems where dimension reduction is successful are Max-Cut, where target dimension $m = 1/\varepsilon^{O(1)}$ suffices [12, 34] (and has immediate implications to streaming algorithms), and projective clustering problems like k-subspace and k-flat approximation, where target dimension that is polynomial in k (but independent of n) suffices [10, 14, 31]. However, the same method cannot get below $m = O(\log n)$ for the k-center problem [29].

This research plan, which may be called "beyond JL", has another thread that seeks bounds that depend on the *intrinsic dimensionality* of the dataset X (instead of n), and specifically on a popular measure called the *doubling dimension*, introduced in [25] based on earlier work by [1, 13]. This notion, denoted $\mathrm{ddim}(X)$, is defined as the minimum $t \geq 0$ such that every ball in X can be covered by at most 2^t balls of half the radius.[2] Observe that $\mathrm{ddim}(X)$ is at most $\log n$ and can often be much smaller, as this notion generalizes Euclidean dimension and can capture many useful cases, like points that lie in a linear subspace or have a sparse vector representation, and even non-Euclidean distances [23, 25].

This line of research aims to show that fundamental problems admit oblivious dimension reduction to dimension $m = m(\varepsilon, \mathrm{ddim}(X))$, and ideally obtain tight bounds. A prime success story is nearest-neighbor search (NNS), for which target dimension $m = \mathrm{ddim}(X)/\varepsilon^{O(1)}$ indeed suffices [28]. However, for three important problems, current results fall short of the above aim: For k-center, the known bound on m has also, i.e., in addition to $\mathrm{ddim}(X)$, an additive term term of $O(\varepsilon^{-2}\log k)$ [29], which seems inevitable. For MST, the known bound on m has also an additive term of $O(\log\log n)$ [41], and this is still open. For uniform facility location (UFL), the known result achieves only $O(1)$-approximation [41], and our main contribution is in fact to significantly improve this approximation factor, from $O(1)$ to $1+\varepsilon$.

Uniform Facility Location (UFL). In this problem, the input is $X \subset \mathbb{R}^d$ and an *opening cost* $\mathsf{f} > 0$, and the goal is to find a set of *facilities* $F \subset \mathbb{R}^d$, so as to minimize the objective

$$\mathrm{cost}(X,F) := \mathsf{f}\cdot|F| + \sum_{x\in X}\mathrm{dist}(x,F),$$

where $\mathrm{dist}(x,F) := \min_{y\in F}\mathrm{dist}(x,y)$ and $\mathrm{dist}(x,y) := \|x-y\|_2$. This is actually a clustering problem very similar to k-median (by viewing facilities as cluster centers), except that the number of clusters $k = |F|$ is not prescribed in advance, which can make the problem easier, as there is no hard constraint on k, but also harder, as bounds cannot depend on k as a parameter. We emphasize that our definition allows facilities to lie in the ambient space, which is natural for a clustering problem (similarly to k-median). Some literature restricts the facilities to a given set, usually the input points, i.e., $F \subset X$, which can make the problem easier, e.g., the algorithm or analysis can enumerate the potential facilities. In contrast, the known dimension reduction for k-median [37] is widely applicable but also technically complicated, precisely because it allows centers to lie in the ambient space.

Remark. A natural approach is to tackle many computational problems at once by refining the JL Lemma so that m would depend on $\mathrm{ddim}(X)$ instead of on n. Unfortunately, this is not possible using linear maps [28, Remark 4.1], which is the method of choice employed in the original JL Lemma. An open question in the area of metric embedding, posed by [25, 35] (see also [39, Question 41]), asks whether every $X \subset \mathbb{R}^d$ embeds in Euclidean space with target dimension and distortion that depend only on $\mathrm{ddim}(X)$ (and not on d or n). Notice that here, the distortion bound is more relaxed and the mapping need not be oblivious or even easy to compute, which would be problematic for algorithmic applications. So far, progress on this open question has been made only for a weaker variant of snowflake embedding [3, 24, 40, 42].

1.1 Results

We study oblivious dimension reduction for inputs that reside in a high-dimension Euclidean space but have a bounded doubling dimension (called in short doubling). Our main result, in Theorem 1.1, achieves $(1+\varepsilon)$-approximation for UFL using target dimension $m = \tilde{O}(\varepsilon^{-2}\mathrm{ddim}(X))$. It uses a map $\pi : \mathbb{R}^d \to \mathbb{R}^m$ that is standard in proofs of the JL Lemma, and is defined by $\pi : x \mapsto \frac{1}{\sqrt{m}}Gx$ where $G \in \mathbb{R}^{m\times d}$ is a random matrix with i.i.d. entries drawn from Gaussian distribution $N(0,1)$. We refer to it as a *random linear map*, although some literature calls it random projection (because it is similar, though not identical, to orthogonal projection onto a random subspace with scaling). Throughout, we assume that the opening cost is $\mathsf{f} = 1$, which holds without loss of generality by rescaling the input $X \subset \mathbb{R}^d$, and denote the optimal value of UFL on input $X \subset \mathbb{R}^d$ by $\mathrm{ufl}(X) := \min\{\mathrm{cost}(X,F) : F \subset \mathbb{R}^d\}$. Let $\mathrm{ddim} \geq 1$ be a known upper bound on the doubling dimension of X, and assume it is given with the input (or in some settings, computed from it).

Theorem 1.1. *Let $0 < \varepsilon, \delta < 1$, let $\mathrm{ddim}, d \geq 1$, and consider a random linear map π with suitable target dimension $m = O(\varepsilon^{-2}\mathrm{ddim}\cdot\log(\delta^{-1}\varepsilon^{-1}\mathrm{ddim}))$. Then for every finite $X \subset \mathbb{R}^d$ with doubling dimension at most ddim,*

$$\Pr[\mathrm{ufl}(\pi(X)) \in (1\pm\varepsilon)\,\mathrm{ufl}(X)] \geq 1-\delta. \tag{2}$$

[1] Throughout, $\tilde{O}(f)$ suppresses factors that are logarithmic in f.
[2] Formally, the centers of these balls must be in X (see Definition 2.1), but relaxing this requirement to center points in the ambient Euclidean space would change $\mathrm{ddim}(X)$ by at most a constant factor.

There are two previous bounds on dimension-reduction for UFL. For $(1+\varepsilon)$-approximation, it was known that dimension $m = \tilde{O}(\varepsilon^{-2}\log n)$ suffices, however when X is doubling our bound is far better. That previous bound follows from dimension reduction for k-median [37], applied with $k = n$, but not from the JL Lemma, because facilities in the ambient space $\mathbb{R}^d$ can evade (1). Another previous result [41] is for $O(1)$-approximation, and shows that dimension $m = O(\mathrm{ddim}(X))$ suffices and is moreover optimal, namely, the map π requires $m = \Omega(\mathrm{ddim}(X))$.[3] We stress here that $O(1)$-approximation for UFL is significantly different from $(1+\varepsilon)$-approximation. In the former, the facilities can be assumed to lie in the dataset X at the cost of factor 2 in the approximation, whereas in the latter, we know of no effective way to discretize the potential facilities in the ambient space $\mathbb{R}^d$, which is truly high-dimensional and does not satisfy the $\mathrm{ddim}(X)$ bound. In a sense, Theorem 1.1 handles a regime that falls between low and high dimension. In fact, the existing tools to tackle this difficulty are quite limited, as in many problems, such as MST, the ambient space is completely irrelevant. Perhaps the closest problem is NNS [28], where query points may come from the ambient space, although the impact of a single query point in NNS is much less global and complicated than facilities in UFL.

It is worthwhile to juxtapose our result with other computational problems. For k-median, dimension reduction is known to require $m = \Omega(\log k)$, even for $O(1)$-approximation of doubling inputs [41], hence we see a sharp contrast with UFL. For MST, which can be viewed as a clustering problem, the known dimension reduction for doubling inputs has an $O(\log\log n)$-term in the target dimension [41], hence our result for UFL may hopefully inspire future improvements.

Our oblivious dimension reduction has immediate implications to offline and streaming algorithms, by simply employing known algorithms for low (Euclidean) dimension. In the offline setting, UFL (and even k-median) in $\mathbb{R}^d$ is known to admit a PTAS, i.e., $(1+\varepsilon)$-approximation for every fixed $\varepsilon > 0$, that runs in time $2^{(1/\varepsilon)^{O(d)}} \cdot n(\log n)^{d+6}$ [32]. Thus, Theorem 1.1 immediately implies $(1+\varepsilon)$-approximation of the optimal value of UFL, on input $X \subset \mathbb{R}^d$ when facilities can lie in the ambient space, in time $2^{(1/\varepsilon)^{\tilde{O}(\mathrm{ddim}(X)/\varepsilon^2)}} \cdot dn(\log n)^{\tilde{O}(\mathrm{ddim}(X)/\varepsilon^2)}$. (We further improve this bound in Theorem 1.3.) We remark that for UFL in doubling metrics (but not necessarily Euclidean), another known algorithm runs in roughly the same time [16], but it restricts the facilities to lie in the dataset X.

In the setting of dynamic geometric streams, the input is a stream of insertions and deletions of points from the grid $[\Delta]^d$, and X is the point set at the end of the stream. One usually assumes that its size is $n \le \mathrm{poly}(\Delta)$, and then bounds can be written in terms of d and Δ (but not n). The known algorithm for this setting uses space $O(\varepsilon^{-1}\log\Delta)^{O(d)}$ and outputs a $(1+\varepsilon)$-approximation to the value $\mathrm{ufl}(X)$ [20].[4] This exponential dependence of d is essential, because in high dimension (which can be reduced to $d = O(\log n)$ because of the JL Lemma), every streaming algorithm that reports an $O(1)$-approximation to $\mathrm{ufl}(X)$ requires $\Omega(\sqrt{n})$ bits of space [19]. Nevertheless, when the doubling dimension of X is low, combining Theorem 1.1 with the algorithm of [20], immediately implies a streaming algorithm that uses significantly less space. As stated below, it essentially decreases the exponent from d to $\mathrm{ddim}(X)/\varepsilon^2$, which can break below the $\mathrm{poly}(n)$ barrier mentioned above [19], e.g., when $\mathrm{ddim}(X) = O(1)$ and $n = \mathrm{poly}(\Delta)$ the space usage is only $\mathrm{polylog}(n)$.

Corollary 1.2. *There is a streaming algorithm that, given as input $0 < \varepsilon < 1$, a set $X \subseteq [\Delta]^d$ presented as a stream of point insertions and deletions, and an upper bound* ddim*, the algorithm uses space $\tilde{O}\big(d \cdot \mathrm{polylog}(\Delta) + (\varepsilon^{-1}\log\Delta)^{\tilde{O}(\mathrm{ddim}/\varepsilon^2)}\big)$ and outputs with high probability a $(1+\varepsilon)$-approximation to* $\mathrm{ufl}(X)$.[5]

PTAS for UFL on Doubling Subsets. Theorem 1.1 only asserts that the optimal *value* is preserved. While it is natural to expect that a solution for UFL on $\pi(X) \subset \mathbb{R}^d$ will yield a solution also for $X \subset \mathbb{R}^m$, formalizing such a connection is tricky, because π is not invertible and there is no natural way to map facilities in $\mathbb{R}^m$ back to $\mathbb{R}^d$.

Nonetheless, we use our dimension reduction in conjunction with a new decomposition procedure that we devise, which partitions a UFL instance $X \subset \mathbb{R}^d$ and effectively reduces it to several k-median instances in $\mathbb{R}^m$, where m is the target dimension from Theorem 1.1 and $k \approx 2^{O(m)}$. This is useful because k-median can be solved efficiently in this parameter regime, for instance, one can use a known PTAS that runs in time $2^{(k/\varepsilon)^{O(1)}} dn$ [33], or alternatively in time $2^{\varepsilon^{-O(d)}} n\log^{d+6} n$ [32]. We thus obtain in Theorem 1.3 the *first* PTAS for UFL on doubling subsets of $\mathbb{R}^d$ where facilities can lie in the ambient space — previous techniques could only handle facilities that are restricted to the dataset X, and we know of no effective way to enumerate the potential facilities in $\mathbb{R}^d$. The entire algorithm is very efficient and runs in near-linear time; it does not even need the input to provide an upper bound ddim, as offline algorithms can $O(1)$-approximate $\mathrm{ddim}(X)$ quickly.

Theorem 1.3. *There is a randomized algorithm that, given as input $0 < \varepsilon < 1$ and an n-point set $X \subset \mathbb{R}^d$, computes a $(1+\varepsilon)$-approximation for UFL in time $(2^{m'}d + 2^{2^{m'}}) \cdot \tilde{O}(n)$ for*

$$m' = O\big(\mathrm{ddim}(X) \cdot \log(\mathrm{ddim}(X)/\varepsilon)\big).$$

Our new decomposition procedure actually works for all doubling metrics (even non-Euclidean). In that setting, it reduces a UFL instance X to several k-median instances in the same metric space (without dimension reduction), for $k \approx 2^{O(\mathrm{ddim}\log(\mathrm{ddim}/\varepsilon))}$. These instances can be solved using known algorithms (based on coresets for k-median) to obtain a PTAS for UFL that runs in time $2^{2^{O(\mathrm{ddim}\cdot\log(\mathrm{ddim}/\varepsilon))}} \cdot \tilde{O}(n)$, when facilities are restricted to the dataset X, and provided oracle access to distances in X. Compared with

[3] Strictly speaking, UFL is defined in [41] with facilities restricted to the input X, but their $O(1)$-approximation applies also in our setting, because one can move the facilities to lie in X at the cost of factor 2. Our $(1+\varepsilon)$-approximation can be adapted also to their setting, see Remark 4.3.

[4] The results in [20] are stated only for $d = 2$, but their analysis seems to extend to every dimension d.

[5] The first term in the space usage is for implementing π, which naively requires $d\log\Delta \cdot \mathrm{ddim}/\varepsilon^2$ bits, using a pseudorandom generator [27], which is now a standard argument. It may be improved further if each stream update is a single coordinate instead of an entire point.

recent work [16] for a similar setting of all doubling metrics,[6] our result improves the dependence on $\mathrm{ddim}(X)$ in the double-exponent from quadratic to near-linear, with comparable dependence on other parameters, e.g., near-linear in n. This expands the recent line of research for pursing fast PTAS for UFL in doubling metrics [16, 21].

1.2 Technical Contributions and Highlights

Our main technical contribution is a new metric decomposition, which partitions a UFL instance that is doubling (not necessarily Euclidean), into multiple instances, each of low value. It has the distinctive feature that facilities can lie in a general ambient space, while previous decompositions require that also the ambient space is doubling. Roughly speaking, our decomposition is a partition $\mathbf{\Lambda}$ of the dataset X into so-called clusters, such that for a suitable parameter $\kappa = (\mathrm{ddim}/\varepsilon)^{\Theta(\mathrm{ddim})}$,

(a) every cluster $C \in \mathbf{\Lambda}$ satisfies $\mathrm{ufl}(C) = \Theta(\kappa)$; and
(b) $\sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(C) \in (1 \pm \varepsilon) \cdot \mathrm{ufl}(X)$.

This decomposition is key to both our dimension-reduction result (Theorem 1.1) and our PTAS (Theorem 1.3). Let us highlight the power of this decomposition. Property (b) guarantees $(1+\varepsilon)$-approximation, which is crucial for surpassing the previous dimension reduction [41], which achieves only $O(1)$-distortion, essentially because it is based on a well-known estimate for $\mathrm{ufl}(X)$, from [38], that provides only $O(1)$-approximation. Property (a) bounds the optimal value of clusters both from below and from above, which is extremely important. Moreover, achieving κ that is independent of n, and specifically $\kappa = (\mathrm{ddim}/\varepsilon)^{\Theta(\mathrm{ddim})}$, is a major strength, because κ determines the target dimension bound, which is actually $O(\log \kappa)$. For comparison, the metric decomposition proposed in [20] achieves $\kappa = \mathrm{polylog}(n)$, which is much weaker, e.g., it would yield dimension reduction with target dimension $O(\log\log n)$, and a QPTAS instead of our PTAS.

Our new decomposition uses a *bottom-up* construction, instead of the previous top-down approach of [20]. Its major advantages is that achieves also a lower bound on $\mathrm{ufl}(C)$, as stated in Property (a), and not only an upper bound that the top-down approach guarantees. This, in turn, is key for achieving κ that is independent of n, because the analysis can charge to the cost of every instance locally. This bottom-up approach is conceptually similar to sparsity decomposition, a technique that was crucial to obtain a PTAS for TSP in doubling metrics [2, 7–9]. That technique employs a bottom-up approach as a preprocessing step to break the dataset into sparse parts that are solved separately, however the UFL problem and the details of our decomposition are completely different.

The terms top-down and bottom-up refer to algorithms that use a hierarchical decomposition of X, which is often randomized. We use Talwar's decomposition [43] for a doubling dataset X, which is analogous to a randomly-shifted quadtree in Euclidean space. Informally, a key feature of this randomized decomposition, denoted by $\mathcal{H}$, is that nearby points are "likely" to be in the same cluster of $\mathcal{H}$ (technically, one considers here a suitably chosen level of $\mathcal{H}$). For UFL, a crucial aspect is whether each data point $x \in X$ is in the same cluster as its nearest facility in a fixed optimal solution F^*, and this creates several challenges. First, an optimal solution F^* is not known to the algorithm (which is not a concern if F^* is needed only in the analysis), and a common workaround is to use instead an $O(1)$-approximate solution F', however it is imperative that the $O(1)$-factor will affect only the additional cost $\varepsilon \cdot \mathrm{ufl}(X)$. Second, facilities that lie in the ambient space are not even part of $\mathcal{H}$, and while conceptually we resolve it similarly to the first challenge, by replacing F^* with proxy near-optimal facilities $F'' \subseteq X$, technically it creates complications in our decomposition and its analysis. Third, even if we restrict the facilities to lie in the dataset X, the guarantees of $\mathcal{H}$ are probabilistic, meaning that some points $x \in X$ (most likely a small fraction) are not in the same cluster with their "optimal" facility, which precludes us from considering that cluster as a separate instance.

An approach proposed in [16] is to eliminate these so-called badly-cut pairs by simply moving each such data point x to its "optimal" facility, effectively creating a modified dataset X' with $\mathrm{ufl}(X') \in (1 \pm \varepsilon)\,\mathrm{ufl}(X)$. This is effective if the subsequent steps are applied to X' with no regard to X, e.g., running a dynamic-programming algorithm on X'. However, for our purpose of decomposing X into low-value clusters (and in turn for our dimension-reduction result), we still need the probabilistic guarantees of $\mathcal{H}$, which apply to X, but not to X' that is derived from that same randomness.

We thus take a different approach of *modifying the hierarchical decomposition* $\mathcal{H}$ instead of the data set X. This step eliminates most, but not all, badly-cut pairs, and we crucially handle the remaining pairs using the probabilistic guarantees of $\mathcal{H}$. We finally construct the partition $\mathbf{\Lambda}$ by employing a bottom-up approach on the (modified) hierarchical decomposition. In principle, each cluster of $\mathbf{\Lambda}$ arises from a cluster in the hierarchical decomposition, however these two clusters are not equal and have a more involved correspondence because of the modifications to $\mathcal{H}$ and the bottom-up approach.

We remark that our decomposition is designed for UFL, however many technical steps are general and may find usage in other problems.

1.3 Proof Overview

As mentioned in Section 1.2, our main technical contribution is a new decomposition for UFL instances, that produces a partition $\mathbf{\Lambda}$ of the dataset X with Properties (a) and (b) from Section 1.2. We provide a technical overview of its construction and proof in Section 1.3.1, and then use this decomposition to prove our dimension-reduction result in Section 1.3.2. Before proceeding, we briefly describe how this decomposition immediately implies a PTAS for UFL.

An Immediate PTAS. With the new decomposition at hand, we can immediately obtain a very efficient PTAS for UFL on a doubling subset $X \subset \mathbb{R}^d$ (the setting of Theorem 1.3): Compute the decomposition $\mathbf{\Lambda}$, and then for each cluster $C \in \mathbf{\Lambda}$, compute a $(1+\varepsilon)$-approximate solution for $\mathrm{ufl}(C)$. To implement the last step, observe that by Property (a), an optimal solution for C opens at most $\mathrm{ufl}(C) \le O(\kappa)$ facilities (recall $\mathfrak{f} = 1$), and thus C can be solved by an algorithm for k-median with $k = O(\kappa)$ (trying also smaller values of k). It suffices to solve k-median within $(1+\varepsilon)$-approximation, which can be done in time $k^{O(k/\varepsilon^3)} \cdot \tilde{O}(n)$ via known approaches

[6] The setting in [16] is slightly more general, where the facilities are restricted to a given *subset* of X, rather than all of X.

based on coresets. By Property (b), the union of these solutions for all $C \in \mathbf{\Lambda}$ is a solution for X that achieves $(1+O(\varepsilon))$-approximation. This PTAS almost matches that of Theorem 1.3, without even using dimension reduction; more precisely, its running time is roughly $2^{2^{O(\mathrm{ddim}\log(\mathrm{ddim}/\varepsilon))}} \cdot \tilde{O}(nd)$, whereas Theorem 1.3 decouples d from the doubly-exponential term, which is significant when d is large, by using our dimension reduction. We refer to the full version [26, Section 5] for implementation details.

1.3.1 New Decomposition Procedure. Our new decomposition for UFL is inspired by an earlier one of [20], although our version is more involved and obtains fundamentally stronger bounds. Let us first recall their approach for an input $X \subset \mathbb{R}^2$. Their procedure applies a randomly-shifted quadtree to partition $X \subset \mathbb{R}^2$, and then scans the quadtree nodes, which correspond to squares in $\mathbb{R}^2$, in a *top-down* manner: When a square C is examined, the procedure tests if $\mathrm{ufl}(C \cap X) \le \kappa$ for a suitable threshold κ. If the test passes, $C \cap X$ is declared as a cluster in the partition $\mathbf{\Lambda}$; otherwise, the procedure is executed recursively on the 4 sub-squares of C. This procedure attains $\mathrm{ufl}(X) \approx \sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(C)$ by a clever charging argument to the parent squares of low-value clusters, but it requires setting $\kappa = \mathrm{polylog}(n)$ (or higher), because the parent squares may be nested and each point inside might be charged $O(\log n)$ times, which originates from the number of levels in the quadtree.

Our decomposition procedure first constructs a randomized hierarchical decomposition $\mathcal{H}$ of X, by applying a standard algorithmic tool, due to Talwar [43], that is analogous to a randomly-shifted quadtree but works for all doubling metrics. This hierarchical decomposition $\mathcal{H}$ has, for every distance scale 2^i, a partition of the dataset X into clusters of diameter at most 2^i, where the partition for each scale 2^{i-1} refines that for 2^i. Moreover, when this $\mathcal{H}$ is viewed as a tree, every cluster has at most $2^{O(\mathrm{ddim})}$ child clusters. The key guarantee of this hierarchical decomposition is the cutting-probability bound

$$\forall x, y \in X, \qquad \Pr[x, y \text{ are in different clusters of scale } 2^i] \le O(\mathrm{ddim}) \cdot \mathrm{dist}(x, y)/2^i. \tag{3}$$

Our decomposition procedure constructs the partition $\mathbf{\Lambda}$ by scanning $\mathcal{H}$ in a *bottom-up* manner, in order to ensure both the upper bound and lower bound in Property (a). (As explained later, we actually use a modified version of $\mathcal{H}$, denoted $\mathcal{T}$.) This is in contrast to the top-down approach of [20], which only guarantees an upper bound on $\mathrm{ufl}(C)$. More precisely, our procedure scans $\mathcal{H}$ bottom-up, starting from the leaf clusters, and processing each cluster only after its child clusters: When a cluster C is examined, and P denotes the current dataset (initialized to X), the procedure tests if $\mathrm{ufl}(C \cap P) \ge \kappa$ for a threshold $\kappa = (\mathrm{ddim}/\varepsilon)^{\Theta(\mathrm{ddim})}$. If the test passes, $C \cap P$ is added as a cluster in $\mathbf{\Lambda}$, the points of C are removed from our current dataset P, and the procedure proceeds to the next cluster in $\mathcal{H}$.

Property (a). The bottom-up construction clearly attains a lower bound $\mathrm{ufl}(C) \ge \kappa$ for all $C \in \mathbf{\Lambda}$ (except for the very last cluster, which we can handle separately). To get an upper bound, observe that a cluster C added to $\mathbf{\Lambda}$ is the union of several child clusters that do not pass the test, i.e., each child C' has $\mathrm{ufl}(C' \cap P) < \kappa$. The number of children is at most $2^{O(\mathrm{ddim})}$, and the union of their optimal solutions is clearly a feasible solution for C, hence $\mathrm{ufl}(C) \le 2^{O(\mathrm{ddim})} \cdot \kappa$. This establishes Property (a), up to relaxing the ratio between the upper and lower bounds to be $2^{O(\mathrm{ddim})}$; the formal treatment appears in Lemma 3.1.

Unfortunately, this bottom-up approach has ramifications that complicate the entire analysis. In particular, a cluster C that is added to $\mathbf{\Lambda}$ is no longer a cluster in the hierarchy $\mathcal{H}$, because some of its descendants in $\mathcal{H}$ might have been removed earlier. This misalignment with $\mathcal{H}$ makes it difficult to use the cutting-probability bound (3), which applies to the clustering in $\mathcal{H}$ but not that in $\mathbf{\Lambda}$. We thus introduce the notion of "holes" (Definition 3.12), which captures the parts of $C \in \mathbf{\Lambda}$ that were removed (when comparing to this C in $\mathcal{H}$). For sake of simplicity, we ignore for now the holes and pretend we are directly analyzing $\mathcal{H}$, and we also ignore the complications arising from the ambient space by assuming that facilities lie inside the dataset X. We will return to discuss these issues later in the section.

Property (b). This property is borrowed from [20], but our proof is completely different, because of the different construction. The high-level idea is to take a set of facilities $F^* \subseteq X$ that is optimal for X, i.e., it realizes $\mathrm{ufl}(X)$, and transform it into a modified set F' by *adding facilities* inside each cluster $C \in \mathbf{\Lambda}$. This F' aligns with our partition $\mathbf{\Lambda}$, because data points in each cluster C are "served locally" by facilities in $F' \cap C$. We will need to show that, in expectation, $\mathrm{cost}(X, F') \le (1+\varepsilon)\,\mathrm{cost}(X, F^*) = (1+\varepsilon)\,\mathrm{ufl}(X)$. To simplify this overview, we present the construction of F' in a more intuitive but less accurate way: Start with $F' = F^*$, then examine each $C \in \mathbf{\Lambda}$ and add to F' a set $N_C \subseteq C$ of extra facilities.

To define this set N_C we need the notion of a net, which is a standard method to discretize a metric space, and is particularly powerful in doubling metrics. Formally, a *ρ-net* of a point set S is a subset $N \subset S$, such that the distance between every two points in N is at least ρ, and every point in S has at least one point of N within distance ρ. Let N_C be an $(\varepsilon' \cdot \mathrm{diam}(C))$-net of C, for $\varepsilon' := \varepsilon/\mathrm{ddim}$; we remark that an $(\varepsilon \cdot \mathrm{diam}(C))$-net may seem sufficient here, however the finer net is needed to compensate for the $O(\mathrm{ddim})$ factor in the cutting-probability bound (3). A standard bound on the size of a net implies that $|N_C| \le O(1/\varepsilon')^{\mathrm{ddim}} = O(\mathrm{ddim}/\varepsilon)^{\mathrm{ddim}}$.

Increase in Cost. We bound the cost increase $\mathrm{cost}(X, F') - \mathrm{cost}(X, F^*)$ by splitting it into two parts, the opening cost and the connection cost. The increases in opening cost of a cluster C is at most $|N_C| \le \varepsilon\kappa \le \varepsilon\,\mathrm{ufl}(C)$ by our choice of κ and Property (a), and in total over all clusters, it is at most $\sum_{C\in\mathbf{\Lambda}} |N_C| \le \varepsilon \sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(C)$, which can be charged to the left-hand side of Property (b), that we shall eventually bound. For the connection cost of each $C \in \mathbf{\Lambda}$, recall that we only use facilities in $C \cap F'$, even though the nearest facility to $x \in C$ might be outside C, and thus the increase in connection cost for C is at most $\Delta_C := \sum_{x\in C} \mathrm{dist}(x, F' \cap C) - \sum_{x\in C} \mathrm{dist}(x, F^*)$.

Now consider $x \in C$ and let $F^*(x)$ be its nearest point in F^*. Observe that if $F^*(x) \in C$ then $\mathrm{dist}(x, F' \cap C) \le \mathrm{dist}(x, F^*)$, i.e., there is no increase, and therefore the nontrivial case is when $F^*(x)$ is outside C. A simple idea is to serve x by its nearest neighbor in N_C, which has connection cost $\mathrm{dist}(x, N_C) \le \varepsilon' \cdot \mathrm{diam}(C)$. However, this bound might be much larger than $\mathrm{dist}(x, F^*)$, and we shall to

eliminate this situation by ensuring a *separation property*:

$$\forall x \in C, \qquad F^*(x) \notin C \implies \mathrm{dist}(x, F^*(x)) \geq \varepsilon' \cdot \mathrm{diam}(C). \quad (4)$$

Indeed, this inequality implies that $\mathrm{dist}(x, N_C) \leq \mathrm{dist}(x, F^*)$, hence serving C by facilities in $F' \cap C$ (instead of F^*) does not increase the connection cost.

Eliminating "Badly Cut" Pairs. We ensure this separation property (4) using the concept of "badly-cut" pairs from [16]. Let $x \in X$, and call a pair $(x, F^*(x))$ *badly cut* if it is cut in the hierarchical decomposition $\mathcal{H}$ at some distance scale $2^i > \mathrm{dist}(x, F^*(x))/\varepsilon'$. Observe that if a pair is not badly cut, then every cluster C in $\mathcal{H}$ that contains x but not $F^*(x)$ must have $\mathrm{diam}(C) \leq \mathrm{dist}(x, F^*(x))/\varepsilon'$. Thus, eliminating all badly-cut pairs ensures the separation property (4).

The badly-cuts pairs are eliminated in [16] by simply moving x to $F^*(x)$ whenever $(x, F^*(x))$ is badly cut. By the cutting-probability bound (3), this happens with probability at most $O(\varepsilon' \cdot \mathrm{ddim}) = O(\varepsilon)$, hence these movements modify X into X' that satisfies $\mathbb{E}_{\mathcal{H}}[\mathrm{ufl}(X')] \in (1 \pm O(\varepsilon)) \cdot \mathrm{ufl}(X)$, which we can afford. The overall strategy here is to first define X' from X, and then add to F^* (which is now the optimal facilities for X') more facilities to obtain F'.

This approach of moving points is effective as a local fix, as it does not change $\mathrm{ufl}(X)$ by too much, however it is not useful for globally decomposing X into clusters that satify Properties (a) and (b). We take an alternative approach of modifying $\mathcal{H}$ (as outlined in Algorithm 2) into a new hierarchical decomposition $\mathcal{T}$, in which no pair $(x, F^*(x))$ is badly cut. We then construct the final partition $\mathbf{\Lambda}$ from this $\mathcal{T}$, rather than from $\mathcal{H}$ (in Algorithm 3). Overall, we establish a refined version of the separation property (4), as detailed in Lemma 3.14.

Handling Holes. Recall that our bottom-up decomposition might create "holes" (Definition 3.12), because a cluster C that is added to $\mathbf{\Lambda}$ might have some of its points removed earlier, and we let Holes_C be the set of clusters in $\mathbf{\Lambda}$ that contain these earlier-deleted points. We can handle holes and still obtain Property (b) using essentially the same arguments as before. When we consider $x \in C$ for some $C \in \mathbf{\Lambda}$ and $F^*(x) \notin C$, we use the net N_C of C (same as before) only when $F^*(x)$ does not belong to any cluster in Holes_C, and we use $N_{\widehat{C}}$ when $F^*(x) \in \widehat{C}$ for some $\widehat{C} \in \mathrm{Holes}_C$. We need to add the nets $N_{\widehat{C}}$ to our opening cost, but this extra cost can be charged to $\mathrm{ufl}(\widehat{C})$, and each $\widehat{C}$ is charged only once by the observation that $\mathrm{Holes}_C \cap \mathrm{Holes}_{C'} = \emptyset$ for distinct $C, C' \in \mathbf{\Lambda}$.

Facilities in the Ambient Space. When facilities can lie in the ambient space, which need not be doubling, we face the major obstacle that the tools we developed, like the cutting-probability bound (3) and the separation property (4) need not apply to the optimal set of facilities $F^* \subset \mathbb{R}^d$. Another, more technical, obstacle is that the net N_C (and similarly $N_{\widehat{C}}$ for $\widehat{C} \in \mathrm{Holes}_C$) might cover the doubling subset C but not $F^*(x)$.

Our plan is to pick for each $F^*(x)$ a *proxy* in the dataset X (which is doubling), adapt our previous arguments to work for that proxy, and use this to argue about F^*. Specifically, the proxy of a facility $F^*(x)$ is its closest point in X that is served (in the optimal solution F^*) by the same facility, formalized by a mapping $g : F^* \to X$, where $g(F^*(x)) = \mathrm{argmin}_{y \in X}\{\mathrm{dist}(y, F^*(x)) : F^*(y) = F^*(x)\}$. To use these proxies, we modify the step that eliminates badly-cut pairs to handle pairs $(x, g(F^*(x)))$ for $x \in X$, and obtain the separation property for these pairs. We then show that this translates also to a separation property for $(x, F^*(x))$.

However, we cannot apply the previous argument about cost increase, because it used that $F^*(x)$ is "covered" by some net N_C (or $N_{\widehat{C}}$ for $\widehat{C} \in \mathrm{Holes}_C$). We need new steps in the analysis, and a particularly nontrivial case is when $g(F^*(x)) \in \widehat{C}$ for some $\widehat{C} \in \mathrm{Holes}_C$. Now, if the proxy $g(F^*(x))$ is close enough to $F^*(x)$, then we can pretend that $F^*(x) = g(F^*(x))$ and the analysis goes through. And if they are far apart (compared with $\mathrm{dist}(x, g(F^*(x)))$), then we crucially make use of the *optimality* of F^*, and show that $F^*(x)$ must be near $\widehat{C}$, namely, within distance $O(\mathrm{diam}(\widehat{C}))$. These facts imply that x is close to $\widehat{C}$, hence x can be covered by the net $N_{\widehat{C}}$. This net is fine enough and thus contains a point within distance $\varepsilon \cdot \mathrm{dist}(x, F^*(x))$ from x (here we use the separation property between $(x, F^*(x))$), and we can use that net point to serve x instead of $F^*(x)$, with no additional connection cost. We remark that these steps generally work for any ambient space beyond Euclidean $\mathbb{R}^d$.

1.3.2 Dimension Reduction. Our proof of dimension reduction for UFL, i.e., that with high probability $\mathrm{ufl}(\pi(X)) \in (1 \pm \varepsilon)\,\mathrm{ufl}(X)$, heavily relies on our decomposition to provide a structurally simple characterization of the optimal value, namely, $\mathrm{ufl}(X) \in (1 \pm \varepsilon) \cdot \sum_{C \in \mathbf{\Lambda}} \mathrm{ufl}(C)$. At a high level, our proof shows that the right-hand side is "preserved" under a random linear map π.

We need to prove both an upper bound and a lower bound on $\mathrm{ufl}(\pi(X))$. The upper bound is easy, as observed in recent work [29, 37, 41], because we may consider one optimal solution F^* for X and analyze its image under π, i.e., the cost of the solution $\pi(F^*)$ for $\pi(X)$. Since we only need π to preserve this one specific solution, target dimension $m = O(\mathrm{poly}(\varepsilon^{-1}))$ suffices.

The lower bound is more interesting and is where we use our decomposition of X, which implies $\mathrm{ufl}(X) \geq (1 - \varepsilon) \cdot \sum_{C \in \mathbf{\Lambda}} \mathrm{ufl}(C)$. We would like to show this inequality is "preserved" under π, i.e., "carries over" to the target space, and what we actually show, as explained further below, is that

$$\mathrm{ufl}(\pi(X)) \geq \sum_{C \in \mathbf{\Lambda}} \mathrm{ufl}(\pi(C)) - \varepsilon \cdot \mathrm{ufl}(X). \quad (5)$$

Notice that the additive error here $\varepsilon \cdot \mathrm{ufl}(X)$ might not be directly comparable to $\mathrm{ufl}(\pi(X))$. Nevertheless, this bound (5) turns out to suffice, because we only need to show in addition that

$$\sum_{C \in \mathbf{\Lambda}} \mathrm{ufl}(\pi(C)) \geq (1 - \varepsilon) \sum_{C \in \mathbf{\Lambda}} \mathrm{ufl}(C). \quad (6)$$

Putting together (5), (6) and Property (b) will then conclude the desired lower bound.

The proof of (6) relies on [37], as follows. Let $\mathrm{med}_k(S)$ denote the optimal value of k-median on $S \subset \mathbb{R}^d$; then we know from [37] that target dimension $\tilde{O}(\varepsilon^{-2} \log k)$ suffices for dimension reduction for k-median, meaning that for every $S \subset \mathbb{R}^d$, with high probability $\mathrm{med}_k(\pi(S)) \in (1 \pm \varepsilon)\,\mathrm{med}_k(S)$. We apply this in our case by letting S be a cluster $C \in \mathbf{\Lambda}$, and we know from Property (a) that the number of facilities needed for C is at most $O(\kappa) = (\mathrm{ddim}/\varepsilon)^{O(\mathrm{ddim})}$, hence target dimension $\tilde{O}(\varepsilon^{-2}\mathrm{ddim})$ suffices for it. The only gap is that we need to apply [37] multiple times for our summation over all

$C \in \mathbf{\Lambda}$. We handle this in a series of lemmas (Lemmas 4.1 and 4.2) that are based on [37], and bound the additive error for each $C \in \mathbf{\Lambda}$ by $\mathrm{ufl}(C) - \mathrm{ufl}(\pi(C)) \le e^{-\varepsilon^2 m} \cdot \mathrm{poly}(\kappa) \le \varepsilon\kappa$. We then use the fact that $|\mathbf{\Lambda}|$, the number of terms in the summation, is roughly $\mathrm{ufl}(X)/\kappa$ (Lemma 3.2), and thus the total additive error is at most $\varepsilon \cdot \mathrm{ufl}(X)$, which we can afford.

Finally, we briefly discuss the proof of (5), which overall is similar to that of Property (b) and its formal treatment appears in Lemma 3.3. We let F^*_π be an optimal set of facilities for $\pi(X)$, and we modify it into F'_π that is "consistent" with $\mathbf{\Lambda}$, i.e., in every cluster $C \in \mathbf{\Lambda}$, all points $x \in \pi(C)$ are served by facilities in $F'_\pi \cap \pi(C)$. Implementing this plan encounters new difficulties, and we focus here on one immediate issue – that we have to analyze $\pi(X)$, which is random. To address this, we condition on the event $\mathcal{E}_C$, for $C \in \mathbf{\Lambda}$, that the distances between points in N_C (which is a net on C) and all other data points (a doubling point set) are preserved simultaneously. For this event to hold with high probability, it suffices that $m = \tilde{O}(\varepsilon^{-2}\mathrm{ddim})$, similarly to a lemma from [28] about preserving the nearest-neighbor distance from a query point to a doubling point set. This is the sole use of the randomness of π in this analysis.

1.4 Related Work

Oblivious dimension reduction can be useful in various models of computation, and one may wonder about algorithms that run in different models and approximate UFL on high-dimensional Euclidean inputs, i.e., inputs as in our results but without the doubling condition. For offline approximation in polynomial time, the state-of-the-art is $(2.406 + \varepsilon)$-approximation for UFL, which follows from the same ratio for k-median [15]. Aiming for fast approximation algorithms, one can achieve $O(1/\varepsilon)$-approximation in time $\tilde{O}(n^{1+\varepsilon})$ [22], via a reduction to nearest neighbor search. This reduction-style result was recently improved to be fully-dynamic, with a similar tradeoff between approximation ratio and time [5]. In dynamic geometric streams, known algorithms achieve $O(d/\log d)$-approximation using $\mathrm{poly}(d \log n)$ space, or $O(1/\varepsilon)$-approximation using space $n^{O(\varepsilon)}\,\mathrm{poly}(d)$, both using a technique of geometric hashing [17]. This geometric-hashing technique was recently used in the setting of massively parallel computing (MPC), to design fully-scalable MPC algorithms that achieve $O(1/\varepsilon)$-approximation in $O(1)$ rounds using $n^{1+\varepsilon}\,\mathrm{poly}(d)$ total space [18].

2 Preliminaries

Let $(\mathcal{X}, \mathrm{dist})$ be a metric space. The *ball* centered at $x \in \mathcal{X}$ with radius $r > 0$ is defined as $B(x, r) := \{y \in \mathcal{X} : \mathrm{dist}(x, y) \le r\}$. The *r-neighborhood* of a point set $X \subseteq \mathcal{X}$ is defined as $B(X, r) := \bigcup_{x \in X} B(x, r)$. The *diameter* of a point set $X \subseteq \mathcal{X}$ is defined as $\mathrm{diam}(X) := \max_{x,y} \mathrm{dist}(x, y)$, and its *aspect ratio* (or *spread*), denoted $\Delta(X)$, is the ratio between the diameter and the minimum inter-point distance in X. For a point set $X \subseteq \mathcal{X}$ and a point $u \in \mathcal{X}$, let $X(u)$ denote the point of X that is nearest to u. Denote by $\mathrm{ufl}^S(X)$ the optimal UFL value for input $X \subseteq \mathcal{X}$ when facilities are restricted to the set $S \subseteq \mathcal{X}$, and let $\mathrm{ufl}(X) := \mathrm{ufl}^{\mathcal{X}}(X)$ for short.

Definition 2.1 (Doubling dimension [25])**.** The *doubling dimension* of a metric space $(\mathcal{X}, \mathrm{dist})$ is the smallest $t \ge 0$ such that every metric ball can be covered by at most 2^t balls of half the radius. The doubling dimension of a point set $X \subseteq \mathcal{X}$ is the doubling dimension of the metric space (X, dist), and is denoted $\mathrm{ddim}(X)$.

Definition 2.2 (Packing, covering and nets)**.** Consider a metric space $(\mathcal{X}, \mathrm{dist})$ and let $\rho > 0$. A point set $S \subseteq \mathcal{X}$ is a *ρ-packing* if for all $x, y \in S$, $\mathrm{dist}(x, y) \ge \rho$. The set S is a *ρ-covering* for X if for every $x \in X$, there is $y \in S$ such that $\mathrm{dist}(x, y) \le \rho$. The set S is a *ρ-net* for X if it is both a ρ-packing and a ρ-covering for X.

Proposition 2.3 (Packing property [25])**.** *If S is ρ-packing then $|S| \le (2\,\mathrm{diam}(S)/\rho)^{\mathrm{ddim}(S)}$.*

We summarize below the properties of the random linear map π are used in this paper. Recall that $\pi : x \mapsto \frac{1}{\sqrt{m}} Gx$ where $G \in \mathbb{R}^{m \times d}$ is a random Gaussian matrix. In some previous work, such as [37], only properties (7) and (9) below were needed, and they may hold for other maps π. We need also (8), which seems to be more specific to Gaussian.

Proposition 2.4 (Properties of random linear maps)**.** *Let $\pi : \mathbb{R}^d \to \mathbb{R}^m$ be a random linear map. Then for every unit vector $x \in \mathbb{R}^d$ and every $t > 0$,*

$$\Pr[\|\pi(x)\| \notin 1 \pm t] \le e^{-t^2 m/8}. \tag{7}$$

$$\Pr[\|\pi(x)\| \le 1/t] \le \left(\frac{3}{t}\right)^m. \tag{8}$$

$$\mathbb{E}\left[\max\{0, \|\pi(x)\| - (1+t)\}\right] \le \frac{1}{mt} e^{-t^2 m/2}. \tag{9}$$

3 A New Decomposition for UFL

This section introduces our new decomposition for UFL instances, which technically is a random partition $\mathbf{\Lambda}$ of the dataset X, and effectively reduces the UFL instance into separate low-value UFL instances, each formed by a different part $C \in \mathbf{\Lambda}$. Throughout this section, we assume that $(\mathcal{X}, \mathrm{dist})$ is an underlying metric space and $X \subseteq \mathcal{X}$ is a dataset of doubling dimension at most ddim. A feasible UFL solution is a set of facilities, which can be any (finite) subset of $\mathcal{X}$. We present the construction of the partition $\mathbf{\Lambda}$ in Section 3.1, which includes a summary of its main properties in Lemmas 3.1 to 3.3. The proofs of these lemmas can be found in the full version [26, Sections 3.2 - 3.4]. The partition $\mathbf{\Lambda}$ is parameterized by $\kappa \ge 1$ (in addition to $0 < \varepsilon < 1$).

Lemma 3.1 (Bounded local UFL values)**.** *For every $\kappa \ge 1$, the random partition $\mathbf{\Lambda} = \mathbf{\Lambda}(\kappa)$ always satisfies that $\kappa \le \mathrm{ufl}(C) \le 2^{10\mathrm{ddim}}\kappa$ for all $C \in \mathbf{\Lambda}$.*

In our applications, we set $\kappa := (\mathrm{ddim}/\varepsilon)^{\Theta(\mathrm{ddim})}$. This ensures that $\mathrm{ufl}(C)$ is small enough for dimension reduction analysis, and in particular an optimal solution $\mathrm{ufl}(C)$ uses at most $2^{10\mathrm{ddim}}\kappa$ facilities, hence finding $\mathrm{ufl}(C)$ reduces to a k-median problem with $k \le 2^{O(\mathrm{ddim})}\kappa$. This is useful in several ways. For instance, a target dimension $m = \tilde{O}(\mathrm{ddim}/\varepsilon^2)$ suffices to preserve $\mathrm{ufl}(\pi(C)) \in (1 \pm \varepsilon)\,\mathrm{ufl}(C)$, via a black-box application of [37], which shows that target dimension $\tilde{O}(\varepsilon^{-2} \log k)$ suffice for k-median. Similarly, as we mentioned, there are also efficient $(1+\varepsilon)$-approximation algorithms for k-median with such small k, which implies a PTAS for $\mathrm{ufl}(C)$.

Lemma 3.2 (Bounding $|\mathbf{\Lambda}|$)**.** *There exist universal constants $c_1, \alpha > 0$, such that for every $\varepsilon \in (0, 1)$ and $\kappa > 2(\mathrm{ddim}/\varepsilon)^{c_1 \cdot \mathrm{ddim}}$, the partition*

$\mathbf{\Lambda} = \mathbf{\Lambda}(\kappa)$ *satisfies*

$$\mathbb{E}\left[|\mathbf{\Lambda}|\right] \le \frac{2\alpha \cdot \mathrm{ufl}(X)}{\kappa - 2(\mathrm{ddim}/\varepsilon)^{c_1 \cdot \mathrm{ddim}}}, \tag{10}$$

where the randomness is over the construction of $\mathbf{\Lambda}$.

Lemma 3.2 essentially says that $|\mathbf{\Lambda}| \le O(\mathrm{ufl}(X)/\kappa)$. This is particularly useful when comparing $\sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(C)$ with $\sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(\pi(C))$ in the dimension-reduction analysis, where we bound the additive error for each $C \in \mathbf{\Lambda}$ by $\mathrm{ufl}(C) - \mathrm{ufl}(\pi(C)) \le \varepsilon\kappa$. Lemma 3.2 then implies that the total additive error is at most $O(\varepsilon) \cdot \mathrm{ufl}(X)$, which we can afford.

We note that the above two lemmas hold for every doubling point set X. The next lemma is specifically for $X \subset \mathbb{R}^d$ (i.e., for the Euclidean metric space $\mathbb{R}^d$), and it analyzes the performance of dimension reduction on $\mathbf{\Lambda}$. This technical lemma provides a lower bound for $\mathrm{ufl}(\pi(X))$ in terms of the local costs $\mathrm{ufl}(\pi(C))$ for $C \in \mathbf{\Lambda}$. This is crucially useful in our dimension reduction analysis.

Lemma 3.3 (Lower bound for $\mathrm{ufl}(\pi(X))$)**.** *Let $\pi\colon \mathbb{R}^d \to \mathbb{R}^m$ be a random linear map, and let $X \subset \mathbb{R}^d$ be finite with doubling dimension at most* ddim*. There exist universal constants $c_1, c_2, c_3 > 0$, such that for every $\varepsilon, \delta \in (0,1)$, if $\kappa > c_2(\mathrm{ddim}/(\delta\varepsilon))^{c_1\cdot\mathrm{ddim}}$ and $m > c_3(\log\kappa + \log(1/\delta\varepsilon))$, then*

$$\Pr\left[\mathrm{ufl}(\pi(X)) \ge \sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(\pi(C)) - \varepsilon\cdot\mathrm{ufl}(X)\right] \ge 1-\delta,$$

where the randomness is over both π and $\mathbf{\Lambda} = \mathbf{\Lambda}(\kappa)$.

In fact, using similar techniques, we can prove a result analogous to this lemma but for general metric $(\mathcal{X}, \mathrm{dist})$ and (finite) doubling subset $X \subseteq \mathcal{X}$, where π is fixed to the identity map.

Corollary 3.4 (Lower bound for $\mathrm{ufl}(X)$)**.** *Let $(\mathcal{X}, \mathrm{dist})$ be a metric space and $X \subseteq \mathcal{X}$ be a finite subset with doubling dimension at most* ddim*. There exist universal constants c_1, c_2, such that for every $\varepsilon, \delta \in (0,1)$ and $\kappa > c_2(\mathrm{ddim}/(\delta\varepsilon))^{c_1\cdot\mathrm{ddim}}$, the random partition $\mathbf{\Lambda} := \mathbf{\Lambda}(\kappa)$ satisfies*

$$\mathrm{ufl}(X) \ge \sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(C) - \varepsilon\cdot\mathrm{ufl}(X), \tag{11}$$

with probability at least $1-\delta$.

3.1 The Construction of $\mathbf{\Lambda}$

Our construction of $\mathbf{\Lambda}$ has three steps. The first one is to compute for X a randomized hierarchical decomposition $\mathcal{H}$, using the algorithm of Talwar [43]. We restate this computation of $\mathcal{H}$ in Algorithm 1, and review its main properties. The second step modifies $\mathcal{H}$ into another hierarchical decomposition $\mathcal{T}$, to eliminate badly-cut pairs (a notion introduced by [16]). As described in Algorithm 2, it works by moving points between clusters separately at each level, and thus each level remains a partition of X, but the nesting across levels might break. The third step constructs the random partition $\mathbf{\Lambda}$ from $\mathcal{T}$, using a bottom-up approach, as described in Algorithm 3. We summarize in Lemma 3.14 several properties of $\mathbf{\Lambda}$ that follow directly from the construction, including a separation and a consistency property, and are essential for proving Lemmas 3.1 to 3.3.

Random Hierarchical Decomposition [43]. We use an algorithm of Talwar [43] to construct a random hierarchical decomposition $\mathcal{H}$, described in Algorithm 1. Let $\gamma := \min\{\mathrm{dist}(x,y) : x \ne y \in X\}$, let $\Delta := \mathrm{diam}(X)/\gamma$ be the aspect ratio of X, and denote $\ell := \lceil \log\Delta \rceil$. At a high level, the algorithm (randomly) partitions X into clusters, and then recursively partitions each cluster into children clusters, where each recursive call decreases the diameter bound by a factor of 2. This process creates a recursion tree, where tree nodes correspond to clusters, and this is referred to as a hierarchical decomposition $\mathcal{H}$. The randomness comes from two sources: (1) the scaling factor ρ, picked in Line 2, which affects the diameter of clusters in Line 6; and (2) the permutation μ, picked in Line 3, which determines the order in which clusters are formed in Line 9. By construction, $\mathcal{H}$ has $\ell+2$ levels. The root node, at the highest level of $\mathcal{H}_{\ell+1}$, corresponds to the trivial cluster X, and each leaf at the lowest level $\mathcal{H}_0$ corresponds to a single point of X. Each node $C \in \mathcal{H}_i$ is the union of all its children at $\mathcal{H}_{i-1}$; see Line 9. Moreover, clusters at every level $\mathcal{H}_i$ form a partition of X, and every cluster $C \in \mathcal{H}_i$ satisfies $\mathrm{diam}(C) \le 2r_i \le 2^i\gamma$. We denote the *diameter-bound* of this cluster by $\overline{\mathrm{diam}}(C) := 2^i\gamma$, and its level by $\mathrm{level}(C) := i$.

Algorithm 1: Random Hierarchical Decomposition [43]

Input: finite point set $X \subset \mathbb{R}^d$ with minimum distance γ and aspect ratio Δ

1 construct nested nets $X = N_0 \supset N_1 \supset \cdots \supset N_\ell$, such that each N_i is a $(2^{i-3}\gamma)$-net of N_{i-1}, where $\ell = \lceil\log\Delta\rceil$
2 pick $\rho \in (\frac{1}{2}, 1)$ uniformly at random
3 pick μ as a random permutation of X
4 $\mathcal{H}_{\ell+1} \leftarrow \{X\}$
5 **for** $i = \ell, \ell-1 \ldots, 0$ **do**
6 $\mathcal{H}_i \leftarrow \emptyset$ and $r_i \leftarrow \rho \cdot 2^{i-1}\gamma$
7 **for** *cluster* $C \in \mathcal{H}_{i+1}$ **do**
8 **for** *each* $y \in N_i$ **do**
9 $C_y \leftarrow C \cap B(y, r_i) \setminus \bigcup_{z\in N_i : \mu(z)<\mu(y)} B(z, r_i)$
 `/* new cluster, a child of C */`
10 $\mathcal{H}_i \leftarrow \mathcal{H}_i \cup \{C_y\}$ `// can skip if` $C_y = \emptyset$
11 **return** $\mathcal{H} \leftarrow \{\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{\ell+1}\}$

We say that a pair $x, \widehat{x} \in X$ is *cut* at level i if there are two distinct clusters $C \ne \widehat{C} \in \mathcal{H}_i$ with $x \in C$ and $\widehat{x} \in \widehat{C}$. We state below a well-known bound on the probability to be cut in $\mathcal{H}$.

Proposition 3.5 (Cutting probability [43])**.** *For every pair $x, \widehat{x} \in X$ and level i,*

$$\Pr[(x,\widehat{x}) \text{ is cut at level } i] \le O\left(\frac{\mathrm{ddim}\cdot\mathrm{dist}(x,\widehat{x})}{2^i\gamma}\right).$$

This bound has been used extensively in previous work, e.g., to argue that nearby points are unlikely to be cut at a high level. We also need the following notion of a badly-cut pair. A similar notion was first introduced in [16], where it is defined with respect to a metric ball, whereas we focus on a pair of points.

Definition 3.6 (Badly-cut pairs)**.** Let $\varepsilon \in (0,1)$. A pair of points $x, \widehat{x} \in X$ is called *ε-badly cut* with respect to $\mathcal{H}$ if $(x, \widehat{x})$ is cut at any level $i \geq \log \frac{\mathrm{ddim} \cdot \mathrm{dist}(x,\widehat{x})}{\varepsilon^2 \gamma}$.

Lemma 3.7 (Badly-cut probability)**.** *Let $\varepsilon \in (0,1)$. Then for every pair $x, \widehat{x} \in X$,*

$$\Pr[(x,\widehat{x}) \text{ is } \varepsilon\text{-badly cut}] \leq O(\varepsilon^2).$$

Proof. Denote $i_0 = \lceil \log \frac{\mathrm{ddim}\, \mathrm{dist}(x,\widehat{x})}{\varepsilon^2 \gamma} \rceil$. By Proposition 3.5,

$$\begin{aligned}\Pr[(x,\widehat{x}) \text{ is } \varepsilon\text{-badly cut}] &\leq \sum_{i \geq i_0} O(\mathrm{ddim}) \cdot 2^{-i}\, \mathrm{dist}(x,\widehat{x})/\gamma \\ &\leq O(\mathrm{ddim}) \cdot 2^{-i_0+1}\, \mathrm{dist}(x,\widehat{x})/\gamma \\ &\leq O(\varepsilon^2). \qquad \square\end{aligned}$$

Fix an α-approximate solution F_0 for the UFL problem on X with $\alpha = O(1)$, such that $F_0 \subseteq X$. (Such a solution always exists by moving the facilities of an optimal solution to their nearest point in the dataset X.) Recall that $F_0(x)$ denotes the closest facility to x in F_0. Our proof examines not only that a pair $(x, \widehat{x})$ is not badly cut, but also that related pairs are not badly cut, as described next.

Definition 3.8 (Good pairs)**.** Let $\varepsilon \in (0,1)$. A pair of points $x, \widehat{x} \in X$ is called *ε-good* with respect to $(\mathcal{H}, F_0)$, if none of the three pairs $(x, \widehat{x})$, $(x, F_0(x))$ and $(\widehat{x}, F_0(\widehat{x}))$ is ε-badly cut with respect to $\mathcal{H}$. When $\varepsilon, \mathcal{H}, F_0$ are clear from the context, we may omit them and simply say that $(x, \widehat{x})$ is good.

The following lemma is an immediate corollary of Lemma 3.7 by the union bound.

Lemma 3.9 (Probability to be good)**.** *Let $\varepsilon \in (0,1)$. Every pair of points $x, \widehat{x} \in X$ (that does not depend on $\mathcal{H}$) is ε-good with probability at least $1 - O(\varepsilon^2)$.*

Our plan is to construct a partition $\mathbf{\Lambda}$ of X so that it has the so-called separation and consistency properties. Informally, the separation property means that for every $x \in X$, if x and $F_0(x)$ belong to different clusters $C \neq \widehat{C} \in \mathbf{\Lambda}$, then $\mathrm{dist}(x, F_0(x))$ is roughly lower bounded by $\Omega(\varepsilon^2/\mathrm{ddim})$ times the maximum of $\overline{\mathrm{diam}}(C)$ and $\overline{\mathrm{diam}}(\widehat{C})$. This property enables us to "represent" a global solution F_0 with respect to some local centers around clusters in $\mathbf{\Lambda}$. Consistency means that every cluster in $\mathbf{\Lambda}$ originates from a cluster in $\mathcal{H}$, and has diameter bound that is not much larger. This property allows us to use a fine net with bounded size as a proxy for candidate centers.

Procedure for Eliminating Badly-Cut Pairs. To achieve the separation property, we need to eliminate all badly-cut pairs. A simple way to eliminate the badly-cut pairs, which was used in [16], is to build a new dataset X' by moving every point $x \in X$ for which $(x, F_0(x))$ is badly cut to the point $F_0(x)$. However, this X' clearly depends on the randomness of $\mathcal{H}$, and thus Proposition 3.5 does not apply to X' (which is actually needed in our subsequent analysis). Hence, we introduce a more sophisticated procedure, in Algorithm 2, that directly modifies the clusters in $\mathcal{H}$ (instead of building a new dataset), and our $\mathbf{\Lambda}$ is then built from the modified decomposition.

The modified decomposition $\mathcal{T}$ is constructed level by level. Initially, $\mathcal{T}$ is a copy of $\mathcal{H}$. Then separately for each level $0 \leq i \leq \ell + 1$, clusters at level i exchange their points in the following way: for every point $x \in X$, if $(x, F_0(x))$ is cut at level i and $i \geq \log \frac{\mathrm{ddim} \cdot \mathrm{dist}(x,F_0(x))}{\varepsilon^2 \gamma}$, then x is moved from its current cluster to the cluster containing $F_0(x)$ (Lines 3-6). Notice that $F_0(x)$ never moves (because $F_0(F_0(x)) = F_0(x)$) and thus the order of processing $x \in X$ does not matter.

Algorithm 2: Modify Decomposition to Eliminate Badly-cut Pairs$(X, \mathcal{H}, F_0, \varepsilon)$

```
1 for i = 0, ..., ℓ + 1 do
2     for each C ∈ H_i, let C^T ← C
3     for x ∈ X do
4         find C, Ĉ ∈ H_i such that x ∈ C and F_0(x) ∈ Ĉ
5         if C ≠ Ĉ and i ≥ log (ddim·dist(x,F_0(x)))/(ε²γ) then
6             let C^T ← C^T \ {x} and Ĉ^T ← Ĉ^T ∪ {x}
7     T_i ← {C^T : C ∈ H_i}   // modified partition of X
8 return T ← {T_0, ..., T_{ℓ+1}}
```

Relation between $\mathcal{T}$ and $\mathcal{H}$. It is easy to see that every level $\mathcal{T}_i \in \mathcal{T}$ still forms a partition of X. We also let $\mathcal{T}$ inherit the tree structure from $\mathcal{H}$, using the one-to-one correspondence between their clusters (ignoring empty clusters), and we write $C^{\mathcal{H}}$ to denote the cluster in $\mathcal{H}$ corresponding to a cluster $C^{\mathcal{T}}$ in $\mathcal{T}$. Observe that now a node $C^{\mathcal{T}} \in \mathcal{T}_i$ *is not* necessarily the union of its children at $\mathcal{T}_{i-1}$. Although the abovementioned one-to-one correspondence exists between $\mathcal{T}$ and $\mathcal{H}$, a significant difference is that an actual cluster $C^{\mathcal{T}} \in \mathcal{T}_i$ need not be the union of all its children in $\mathcal{T}_{i-1}$.

Properties of $\mathcal{T}$. We can reinterpret Definition 3.6 of badly-cut pairs with respect to $\mathcal{T}$ (recall it was originally defined with respect to $\mathcal{H}$): A pair $(x, \widehat{x})$ is *ε-badly cut* with respect to $\mathcal{T}$ if there exists a level $i \geq \log \frac{\mathrm{ddim} \cdot \mathrm{dist}(x,F_0(x))}{\varepsilon^2 \gamma}$ and different clusters $C^{\mathcal{T}} \neq \widehat{C}^{\mathcal{T}} \in \mathcal{T}_i$, such that $x \in C^{\mathcal{T}}$ and $\widehat{x} \in \widehat{C}^{\mathcal{T}}$. Observe that once x is moved to the cluster containing $F_0(x)$ at level i, then it always stays in the same cluster as $F_0(x)$ at higher levels $j \geq i$. Hence, the next fact follows immediately from the steps of Algorithm 2.

Fact 3.10. *Every pair $(x, F_0(x))$ for $x \in X$ is not badly cut with respect to $\mathcal{T}$.*

The next lemma shows that $\mathcal{T}$ maintains consistency with $\mathcal{H}$, i.e., the diameter of each cluster $C^{\mathcal{T}}$ does not exceed that of $C^{\mathcal{H}}$ by much. Recall that $\overline{\mathrm{diam}}(C^{\mathcal{H}}) = 2^i \gamma$ for all $C^{\mathcal{H}} \in \mathcal{H}_i$, and that for a point set Y and $r > 0$, we denote $B(Y, r) = \bigcup_{x \in Y} B(x, r)$.

Lemma 3.11 (Consistency of $\mathcal{T}$)**.** *Let $\varepsilon \in (0,1)$ and $\mathcal{T} = \mathcal{T}(X, \mathcal{H}, F_0, \varepsilon)$ be constructed by Algorithm 2. Then for every $i \in \{0, 1, \ldots, \ell+1\}$ and cluster $C^{\mathcal{T}} \in \mathcal{T}_i$, it holds that $C^{\mathcal{T}} \subseteq B(C^{\mathcal{H}}, \varepsilon^2 \cdot 2^i \gamma)$, and thus $C^{\mathcal{T}} \subseteq B(C^{\mathcal{H}}, \varepsilon^2\, \overline{\mathrm{diam}}(C^{\mathcal{H}}))$.*

Proof. For every point $x \in C^{\mathcal{T}} \setminus C^{\mathcal{H}}$, $F_0(x) \in C^{\mathcal{H}}$ and $i \geq \log \frac{\mathrm{ddim} \cdot \mathrm{dist}(x,F_0(x))}{\varepsilon^2 \gamma}$. Hence, $\mathrm{dist}(x, F_0(x)) \leq \frac{\varepsilon^2\, \overline{\mathrm{diam}}(C^{\mathcal{H}})}{\mathrm{ddim}} \leq \varepsilon^2\, \overline{\mathrm{diam}}(C^{\mathcal{H}})$. This completes the proof. $\square$

Constructing the Partition $\mathbf{\Lambda}$. We can now present Algorithm 3 the construction of $\mathbf{\Lambda}$, which works in a bottom-up manner, as

follows. Given a threshold $\kappa > 0$, we find the lowest-level cluster C in $\mathcal{T}$ such that $\mathrm{ufl}(C) \geq \kappa$, and add it to the partition $\mathbf{\Lambda}$ (Lines 2 and 4). We then remove the points of C from X and from every cluster in $\mathcal{T}$ (Line 5). We repeat this procedure until all points in X are removed, or not suitable C exists, in which case we simply add the remaining points in X as a separate part (Line 7). It is easy to see that the output $\mathbf{\Lambda}$ forms a partition of X.

We remark that the last cluster C added to $\mathbf{\Lambda}$ might have $\mathrm{ufl}(C^{\mathcal{T}}) < \kappa$, which violates Lemma 3.1. This special cluster does not affect the correctness of Lemmas 3.2 and 3.3 and thus for simplicity, we assume that all clusters $C \in \mathbf{\Lambda}$ satisfy $\mathrm{ufl}(C) \geq \kappa$. To remove this assumption, we can also merge the last two clusters added to $\mathbf{\Lambda}$, as it would violate the upper bound on $\mathrm{ufl}(C)$ by at most factor 2.

Algorithm 3: PARTITION$(X, \mathcal{T}, \kappa)$

```
1 while X ≠ ∅ do
2     let 0 ≤ i ≤ ℓ be the smallest integer such that there is
        C ∈ T_i with ufl(C) ≥ κ
3     if such i, C exist then
4         Λ ← Λ ∪ {C}
5         X ← X \ C, and update for every j all clusters
            Ĉ ∈ T_j by Ĉ ← Ĉ \ C
6     else
7         Λ ← Λ ∪ {X}                    // add last cluster
8         X ← ∅
9 return Λ
```

Relation between $\mathbf{\Lambda}$ and $\mathcal{T}$. Recall that there is a one-to-one correspondence between clusters in $\mathcal{T}$ and $\mathcal{H}$. We can define a relation also between clusters in $\mathbf{\Lambda}$ and in $\mathcal{T}$ (and hence in $\mathcal{H}$), by tracking the steps in Algorithm 3. Specifically, a cluster $C \in \mathbf{\Lambda}$ is usually added to $\mathbf{\Lambda}$ in Line 4, so there is a clearly defined correspondence with this cluster C in $\mathcal{T}$. In the exceptional case of the last cluster, added in Line 7), C contains all remaining points so we can define its corresponding cluster in $\mathcal{T}$ to be the root, which is the entire dataset X. For a part $C \in \mathbf{\Lambda}$, we write $C^{\mathcal{T}}$ and $C^{\mathcal{H}}$ to denote its corresponding clusters in $\mathcal{T}$ and $\mathcal{H}$.

Due to the bottom-up nature of the construction of $\mathbf{\Lambda}$, clusters $C \in \mathbf{\Lambda}$ may not be perfectly aligned with its corresponding cluster $C^{\mathcal{T}} \in \mathcal{T}$. To see this, consider a cluster $C \in \mathbf{\Lambda}$, and suppose another cluster $\widehat{C} \in \mathbf{\Lambda}$ was added to $\mathbf{\Lambda}$ before C during the execution of Algorithm 3. If $\widehat{C}^{\mathcal{T}}$ is a descendant of $C^{\mathcal{T}}$, then we must remove $\widehat{C}$ from $C^{\mathcal{T}}$ when constructing $\mathbf{\Lambda}$, which makes the cluster C a subset of $C^{\mathcal{T}} \setminus \widehat{C}$ (instead of a full cluster in $\mathcal{T}$). Thus, we observe that $C \subseteq C^{\mathcal{T}}$ holds for every $C \in \mathbf{\Lambda}$. Next, we define the following structure called *holes* for clusters $C \in \mathbf{\Lambda}$ to capture such misalignment between C and $C^{\mathcal{T}}$.

Definition 3.12 (Holes). A cluster $\widehat{C} \in \mathbf{\Lambda}$ is called a *hole* of $C \in \mathbf{\Lambda}$ if among all clusters in $\mathbf{\Lambda}$, C is the one whose corresponding $C^{\mathcal{T}}$ is the lowest-level ancestor of $\widehat{C}^{\mathcal{T}}$ (in $\mathcal{T}$). The set of holes of $C \in \mathbf{\Lambda}$ is defined as $\mathrm{Holes}_C := \{\widehat{C} \in \mathbf{\Lambda} : \widehat{C} \text{ is a hole of } C\}$.

Lemma 3.13 (Total number of holes). $\sum_{C \in \mathbf{\Lambda}} |\mathrm{Holes}_C| \leq |\mathbf{\Lambda}|$.

PROOF. By definition, each $\widehat{C} \in \mathbf{\Lambda}$ is a hole of at most one C, i.e., $\mathrm{Holes}_C \cap \mathrm{Holes}_{C'} = \emptyset$ for distinct $C, C' \in \mathbf{\Lambda}$. Therefore, the total size of all Holes_C is upper bounded by the size of $\mathbf{\Lambda}$. □

Finally, the following lemma summarizes the desired properties of $\mathbf{\Lambda}$, which are useful for dimension-reduction analysis.

Lemma 3.14. *Consider a random partition $\mathbf{\Lambda} = \mathbf{\Lambda}(X, \mathcal{T}, \kappa)$.*

(1) ***Separation****: For every ε-good pair $(x, \widehat{x})$ with respect to $(\mathcal{H}, F_0)$ with $x \in C$, $\widehat{x} \in \widehat{C}$ and $C \neq \widehat{C} \in \mathbf{\Lambda}$, the following holds.*
  (a) If $C^{\mathcal{H}}$ and $\widehat{C}^{\mathcal{H}}$ are not related (as descendant-ancestor) in $\mathcal{H}$, then $\mathrm{dist}(x, \widehat{x}) \geq \frac{\varepsilon^2}{\mathrm{ddim}} \cdot \max\{\overline{\mathrm{diam}}(C^{\mathcal{H}}), \overline{\mathrm{diam}}(\widehat{C}^{\mathcal{H}})\}$.
  (b) If $\widehat{C}^{\mathcal{H}}$ is a descendant of $C^{\mathcal{H}}$ in $\mathcal{H}$, then there exists a cluster $\widetilde{C} \in \mathrm{Holes}_C$, such that $\mathrm{dist}(x, \widehat{x}) \geq \frac{\varepsilon^2}{\mathrm{ddim}} \cdot \overline{\mathrm{diam}}(\widetilde{C}^{\mathcal{H}})$.

(2) ***Consistency****: For every cluster $C \in \mathbf{\Lambda}$, it holds that $C \subseteq B(C^{\mathcal{H}}, \varepsilon^2 \overline{\mathrm{diam}}(C^{\mathcal{H}}))$.*

Let us explain the separation property of Lemma 3.14. Case (1a) is more intuitive, because if $C^{\mathcal{H}}$ and $\widehat{C}^{\mathcal{H}}$ are not related in $\mathcal{H}$ (related means that one is ancestor of the other), then $C^{\mathcal{H}} \cap \widehat{C}^{\mathcal{H}} = \emptyset$, which implies the distance lower bound. However, in case (1b), $\widehat{C}^{\mathcal{H}}$ is a subset of $C^{\mathcal{H}}$, meaning that $\mathrm{dist}(\widehat{x}, C^{\mathcal{H}}) = 0$, which corresponds to the misalignment in $\mathbf{\Lambda}$ discussed earlier. We thus need to use Holes_C to obtain the separation property, which is a major structural complication for our bottom-up construction of $\mathbf{\Lambda}$. In particular, in our later arguments where we wish to find a net N_C for $C \in \mathbf{\Lambda}$ whose granularity depends on the separation guarantee in Lemma 3.14, we not only need a net for C but also a series of nets on clusters in Holes_C. See the full version [26, Sections 3.3, 3.4] for details.

PROOF OF LEMMA 3.14. We first show the separation property. By the definition of ε-good pairs (Definition 3.8), neither of x and $\widehat{x}$ is moved to another cluster during the execution of Algorithm 2. Thus $x \in C^{\mathcal{H}}$ and $\widehat{x} \in \widehat{C}^{\mathcal{H}}$.

If $C^{\mathcal{H}}$ and $\widehat{C}^{\mathcal{H}}$ are not related in $\mathcal{H}$ (related means that one is ancestor of the other), then x and $\widehat{x}$ are cut at level $\max\{\mathrm{level}(C^{\mathcal{H}}), \mathrm{level}(\widehat{C}^{\mathcal{H}})\}$ of $\mathcal{H}$. Since $(x, \widehat{x})$ is not ε-badly cut with respect to $\mathcal{H}$, we have $\max\{\mathrm{level}(C^{\mathcal{H}}), \mathrm{level}(\widehat{C}^{\mathcal{H}})\} \leq \log \frac{\mathrm{ddim} \cdot \mathrm{dist}(x, \widehat{x})}{\varepsilon^2 \gamma}$, or equivalently, $\mathrm{dist}(x, \widehat{x}) \geq \frac{\varepsilon^2}{\mathrm{ddim}} \max\{\overline{\mathrm{diam}}(C^{\mathcal{H}}), \overline{\mathrm{diam}}(\widehat{C}^{\mathcal{H}})\}$.

If $\widehat{C}^{\mathcal{H}}$ is a descendant of $C^{\mathcal{H}}$ in $\mathcal{H}$, then there exists $\widetilde{C} \in \mathrm{Holes}_C$, such that $\widetilde{C}^{\mathcal{H}}$ is a descendant of $C^{\mathcal{H}}$ and an ancestor of $\widehat{C}^{\mathcal{H}}$, and that $(x, \widehat{x})$ is cut at level $\mathrm{level}(\widetilde{C}^{\mathcal{H}})$ of $\mathcal{H}$. Since $(x, \widehat{x})$ is not ε-badly cut with respect to $\mathcal{H}$, we have $\mathrm{level}(\widetilde{C}^{\mathcal{H}}) \leq \log \frac{\mathrm{ddim} \cdot \mathrm{dist}(x, \widehat{x})}{\varepsilon^2 \gamma}$, or equivalently, $\mathrm{dist}(x, \widehat{x}) \geq \frac{\varepsilon^2}{\mathrm{ddim}} \overline{\mathrm{diam}}(\widetilde{C}^{\mathcal{H}})$.

Finally, observe that $C \subseteq C^{\mathcal{T}}$, hence the consistency of $\mathbf{\Lambda}$ follows immediately from the consistency of $\mathcal{T}$ (Lemma 3.11). □

4 Proof of Theorem 1.1: Dimension Reduction for UFL

Theorem 1.1. *Let $0 < \varepsilon, \delta < 1$, let $\mathrm{ddim}, d \geq 1$, and consider a random linear map π with suitable target dimension $m = O(\varepsilon^{-2}\mathrm{ddim} \cdot \log(\delta^{-1}\varepsilon^{-1}\mathrm{ddim}))$. Then for every finite $X \subset \mathbb{R}^d$ with doubling*

dimension at most ddim,

$$\Pr[\mathrm{ufl}(\pi(X)) \in (1 \pm \varepsilon)\,\mathrm{ufl}(X)] \geq 1 - \delta. \tag{2}$$

We start by stating two technical lemmas. First, we have the following lemma that upper bounds the expansion of $\mathrm{ufl}(\pi(X))$. This lemma is essential for the proof of Theorem 1.1, where we apply it directly on X to obtain an upper bound. The proof of Lemma 4.1 can be found in the full version [26, Appendix B.1].

Lemma 4.1 (An upper bound of $\mathrm{ufl}(\pi(X))$)**.** *Let $X \subset \mathbb{R}^d$ be a finite point set. Let $\pi\colon \mathbb{R}^d \to \mathbb{R}^m$ be a random linear map. Then for every $t > 0$,*

$$\mathbb{E}\left[\max\{0, \mathrm{ufl}(\pi(X)) - (1+t)\,\mathrm{ufl}(X)\}\right] \leq \frac{1}{mt} e^{-t^2 m/2}\,\mathrm{ufl}(X).$$

Furthermore,

$$\Pr\left[\mathrm{ufl}(\pi(X)) \geq (1+t)\,\mathrm{ufl}(X)\right] \leq \frac{4}{t^2 m} e^{-t^2 m/8}.$$

We also conclude the following lemma from [37] to control the contraction of each $\mathrm{ufl}(C)$, which is useful for relating $\sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(\pi(C))$ and $\sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(C)$ in the proof of Theorem 1.1. The proof of Lemma 4.2 can be found in the full version [26, Appendix B.2].

Lemma 4.2. *Let $C \subset \mathbb{R}^d$ be a finite point set with $\mathrm{ufl}(C) \leq \tau$. Let $\pi\colon \mathbb{R}^d \to \mathbb{R}^m$ be a random linear map. Then there exists a universal constant $c > 0$, such that for every $\varepsilon \in (0,1)$, if $m > c \cdot \varepsilon^{-2}\log(1/\varepsilon)$, then*

$$\Pr\left[\mathrm{ufl}(\pi(C)) \leq \frac{1}{1+\varepsilon}\,\mathrm{ufl}(C)\right] \leq \tau^3 \cdot e^{-\Omega(\varepsilon^2 m)}.$$

Recall that our typical choice of the parameter is $\tau = (\mathrm{ddim}/\varepsilon)^{O(\mathrm{ddim})}$. Thus, a target dimension $m = O(\varepsilon^{-2}\mathrm{ddim}\log(\mathrm{ddim}/\varepsilon))$ suffices to bound the expected contraction on C within ε, which achieves the target dimension bound in Theorem 1.1.

Proof of Theorem 1.1. Noting that $m = \Omega(\varepsilon^{-2}\log(1/(\delta\varepsilon)))$, the desired upper bound of $\mathrm{ufl}(\pi(X))$, i.e. $\Pr[\mathrm{ufl}(\pi(X)) \leq (1+\varepsilon)\,\mathrm{ufl}(X)] \geq 1 - \delta/2$ follows immediately from Lemma 4.1.

Now we turn to the lower bound of $\mathrm{ufl}(\pi(X))$. Let parameter $\kappa := c_2(\mathrm{ddim}/(\delta\varepsilon))^{c_1\cdot\mathrm{ddim}}$ satisfy the condition in Lemma 3.3. Let $\mathbf{\Lambda} := \mathbf{\Lambda}(\kappa)$ be the random partition constructed in Section 3. By Lemma 3.1, $\kappa \leq \mathrm{ufl}(C) \leq 2^{10\mathrm{ddim}}\kappa$ holds for every $C \in \mathbf{\Lambda}$. Denote $\tau := 2^{10\mathrm{ddim}}\kappa$ to be an upper bound for every $\mathrm{ufl}(C)$. We choose $m = c\cdot\varepsilon^{-2}(\log\tau + \log(1/\delta\varepsilon)) = O(\varepsilon^{-2}\mathrm{ddim}(\log\mathrm{ddim} + \log(1/\delta\varepsilon)))$, where c is a sufficiently large constant.

We start from relating each $\mathrm{ufl}(\pi(C))$ to $\mathrm{ufl}(C)$. Conditioning on the randomness of $\mathcal{H}$,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\pi}\left[\max\{0, (1-\varepsilon/3)\,\mathrm{ufl}(C) - \mathrm{ufl}(\pi(C))\} \mid \mathcal{H}\right] \\
&\leq \mathrm{ufl}(C)\cdot\Pr_{\pi}\left[\mathrm{ufl}(\pi(C)) \leq (1-\varepsilon/3)\,\mathrm{ufl}(C) \mid \mathcal{H}\right] \\
&\leq \tau\cdot\Pr_{\pi}\left[\mathrm{ufl}(\pi(C)) \leq \frac{1}{1+\varepsilon/3}\,\mathrm{ufl}(C) \mid \mathcal{H}\right] \\
&\leq \tau^4\cdot e^{-\Omega(\varepsilon^2 m)}. && \text{(Lemma 4.2)}
\end{aligned}$$

Summing over all $C \in \mathbf{\Lambda}$, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\pi,\mathcal{H}}\left[\sum_{C\in\mathbf{\Lambda}} \max\{0, (1-\varepsilon/3)\,\mathrm{ufl}(C) - \mathrm{ufl}(\pi(C))\}\right] \\
&\leq \tau^4\cdot e^{-\Omega(\varepsilon^2 m)}\cdot\mathop{\mathbb{E}}_{\mathcal{H}}[|\mathbf{\Lambda}|] \\
&\leq \tau^4\cdot e^{-\Omega(\varepsilon^2 m)}\cdot\frac{2\alpha\,\mathrm{ufl}(X)}{\kappa - 2(\mathrm{ddim}/\varepsilon)^{O(\mathrm{ddim})}} && \text{(Lemma 3.2)} \\
&\leq \delta\varepsilon^2/6\cdot\mathrm{ufl}(X).
\end{aligned}$$

By Markov's inequality, with probability at least $1 - \delta/2$,

$$\begin{aligned}
\sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(\pi(C)) &\geq (1-\varepsilon/3)\sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(C) - \varepsilon^2/3\,\mathrm{ufl}(X) \\
&\geq (1 - 2\varepsilon/3)\,\mathrm{ufl}(X). && (12)
\end{aligned}$$

On the other hand, by Lemma 3.3, with probability at least $1-\delta/2$,

$$\mathrm{ufl}(\pi(X)) \geq \sum_{C\in\mathbf{\Lambda}} \mathrm{ufl}(\pi(C)) - \varepsilon/3\cdot\mathrm{ufl}(X). \tag{13}$$

Combining (13) and (12), with probability at least $1 - \delta$,

$$\mathrm{ufl}(\pi(X)) \geq (1-\varepsilon)\,\mathrm{ufl}(X),$$

which completes the proof. □

Remark 4.3. Recall that $\mathrm{ufl}^S(X)$ stands for the optimal UFL value on X subject to the constraint that the facilities must be taken from S, defined in Section 2. Using a variant of Lemma 4.2, we can prove the same target-dimension bound for the *discrete setting*, i.e.,

$$\Pr\left[\mathrm{ufl}^{\pi(X)}(\pi(X)) \in (1\pm\varepsilon)\,\mathrm{ufl}^X(X)\right] \geq 1 - \delta,$$

which directly improves over the $O(1)$-approximate of [41].

References

[1] Patrice Assouad. 1983. Plongements lipschitziens dans $\mathbb{R}^n$. *Bull. Soc. Math. France* 111, 4 (1983), 429–448. doi:10.24033/bsmf.1997

[2] Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. 2016. The Traveling Salesman Problem: Low-Dimensionality Implies a Polynomial Time Approximation Scheme. *SIAM J. Comput.* 45, 4 (2016), 1563–1581. doi:10.1137/130913328

[3] Yair Bartal, Ben Recht, and Leonard J. Schulman. 2011. Dimensionality reduction: Beyond the Johnson-Lindenstrauss bound. In *SODA*. SIAM, 868–887. doi:10.1137/1.9781611973082.68

[4] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. 2019. Oblivious dimension reduction for k-means: beyond subspaces and the Johnson-Lindenstrauss lemma. In *STOC*. ACM, 1039–1050. doi:10.1145/3313276.3316318

[5] Sayan Bhattacharya, Gramoz Goranci, Shaofeng H.-C. Jiang, Yi Qian, and Yubo Zhang. 2024. Dynamic Facility Location in High Dimensional Euclidean Spaces. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=rucbIsWoEV

[6] Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. 2010. Random Projections for k-means Clustering. In *NIPS*. Curran Associates, Inc., 298–306. https://proceedings.neurips.cc/paper/2010/hash/73278a4a86960eeb576a8fd4c9ec6997-Abstract.html

[7] T.-H. Hubert Chan, Shuguang Hu, and Shaofeng H.-C. Jiang. 2018. A PTAS for the Steiner Forest Problem in Doubling Metrics. *SIAM J. Comput.* 47, 4 (2018), 1705–1734. doi:10.1137/16M1107206

[8] T.-H. Hubert Chan, Haotian Jiang, and Shaofeng H.-C. Jiang. 2020. A Unified PTAS for Prize Collecting TSP and Steiner Tree Problem in Doubling Metrics. *ACM Trans. Algorithms* 16, 2 (2020), 24:1–24:23. doi:10.1145/3378571

[9] T.-H. Hubert Chan and Shaofeng H.-C. Jiang. 2018. Reducing Curse of Dimensionality: Improved PTAS for TSP (with Neighborhoods) in Doubling Metrics. *ACM Trans. Algorithms* 14, 1 (2018), 9:1–9:18. doi:10.1145/3158232

[10] Moses Charikar and Erik Waingarten. 2025. The Johnson-Lindenstrauss Lemma for Clustering and Subspace Approximation: From Coresets to Dimension Reduction. In *SODA*. SIAM, 3172–3209. doi:10.1137/1.9781611978322.102 arXiv:2205.00371

[11] Xi Chen, Vincent Cohen-Addad, Rajesh Jayaram, Amit Levi, and Erik Waingarten. 2023. Streaming Euclidean MST to a Constant Factor. In *STOC*. ACM, 156–169. doi:10.1145/3564246.3585168

[12] Xiaoyu Chen, Shaofeng H.-C. Jiang, and Robert Krauthgamer. 2023. Streaming Euclidean Max-Cut: Dimension vs Data Reduction. In *STOC*. ACM, 170–182. doi:10.1145/3564246.3585170

[13] Kenneth L. Clarkson. 1999. Nearest Neighbor Queries in Metric Spaces. *Discret. Comput. Geom.* 22, 1 (1999), 63–93. doi:10.1007/PL00009449

[14] Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. 2015. Dimensionality Reduction for k-Means Clustering and Low Rank Approximation. In *STOC*. ACM, 163–172. doi:10.1145/2746539.2746569

[15] Vincent Cohen-Addad, Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. 2022. Improved approximations for Euclidean k-means and k-median, via nested quasi-independent sets. In *STOC*. ACM, 1621–1628. doi:10.1145/3519935.3520011

[16] Vincent Cohen-Addad, Andreas Emil Feldmann, and David Saulpic. 2021. Near-linear Time Approximation Schemes for Clustering in Doubling Metrics. *J. ACM* 68, 6 (2021), 44:1–44:34. doi:10.1145/3477541

[17] Artur Czumaj, Arnold Filtser, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselỳ, and Mingwei Yang. 2022. Streaming Facility Location in High Dimension via Geometric Hashing. *CoRR* (2022). arXiv:2204.02095 The latest version has additional results compared to the preliminary version in [19].

[18] Artur Czumaj, Guichen Gao, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Pavel Veselý. 2024. Fully-Scalable MPC Algorithms for Clustering in High Dimension. In *ICALP (LIPIcs, Vol. 297)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 50:1–50:20. doi:10.4230/LIPIcs.ICALP.2024.50

[19] Artur Czumaj, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselý, and Mingwei Yang. 2022. Streaming Facility Location in High Dimension via Geometric Hashing. In *FOCS*. IEEE, 450–461. doi:10.1109/FOCS54457.2022.00050

[20] Artur Czumaj, Christiane Lammersen, Morteza Monemizadeh, and Christian Sohler. 2013. $(1+\varepsilon)$-approximation for facility location in data streams. In *SODA*. SIAM, 1710–1728. doi:10.1137/1.9781611973105.123

[21] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. 2019. Local Search Yields a PTAS for k-Means in Doubling Metrics. *SIAM J. Comput.* 48, 2 (2019), 452–480. doi:10.1137/17M1127181

[22] Ashish Goel, Piotr Indyk, and Kasturi R. Varadarajan. 2001. Reductions among high dimensional proximity problems. In *SODA*. ACM/SIAM, 769–778. http://dl.acm.org/citation.cfm?id=365411.365776

[23] Lee-Ad Gottlieb, Aryeh Kontorovich, and Robert Krauthgamer. 2014. Efficient Classification for Metric Data. *IEEE Trans. Inf. Theory* 60, 9 (2014), 5750–5759. doi:10.1109/TIT.2014.2339840

[24] Lee-Ad Gottlieb and Robert Krauthgamer. 2015. A Nonlinear Approach to Dimension Reduction. *Discret. Comput. Geom.* 54, 2 (2015), 291–315. doi:10.1007/s00454-015-9707-9

[25] Anupam Gupta, Robert Krauthgamer, and James R. Lee. 2003. Bounded Geometries, Fractals, and Low-Distortion Embeddings. In *FOCS*. IEEE Computer Society, 534–543. doi:10.1109/SFCS.2003.1238226

[26] Lingxiao Huang, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Di Yue. 2024. Near-Optimal Dimension Reduction for Facility Location. *CoRR* (2024). arXiv:2411.05432

[27] Piotr Indyk. 2006. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53, 3 (2006), 307–323. doi:10.1145/1147954.1147955

[28] Piotr Indyk and Assaf Naor. 2007. Nearest-neighbor-preserving embeddings. *ACM Trans. Algorithms* 3, 3 (2007), 31. doi:10.1145/1273340.1273347

[29] Shaofeng H.-C. Jiang, Robert Krauthgamer, and Shay Sapir. 2024. Moderate Dimension Reduction for k-Center Clustering. In *SoCG (LIPIcs, Vol. 293)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 64:1–64:16. doi:10.4230/LIPIcs.SoCG.2024.64

[30] William Johnson and Joram Lindenstrauss. 1984. Extensions of Lipschitz maps into a Hilbert space. *Contemp. Math.* 26 (01 1984), 189–206. doi:10.1090/conm/026/737400

[31] Michael Kerber and Sharath Raghvendra. 2015. Approximation and Streaming Algorithms for Projective Clustering via Random Projections. In *CCCG*. Queen's University, Ontario, Canada. http://research.cs.queensu.ca/cccg2015/CCCG15-papers/16.pdf

[32] Stavros G. Kolliopoulos and Satish Rao. 2007. A Nearly Linear-Time Approximation Scheme for the Euclidean k-Median Problem. *SIAM J. Comput.* 37, 3 (2007), 757–782. doi:10.1137/S0097539702404055

[33] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. 2010. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM* 57, 2 (2010). doi:10.1145/1667053.1667054

[34] Christiane Lammersen, Anastasios Sidiropoulos, and Christian Sohler. 2009. Streaming Embeddings with Slack. In *WADS (Lecture Notes in Computer Science, Vol. 5664)*. Springer, 483–494. doi:10.1007/978-3-642-03367-4_42

[35] Urs Lang and Conrad Plaut. 2001. Bilipschitz embeddings of metric spaces into space forms. *Geometriae Dedicata* 87, 1-3 (2001), 285–307. doi:10.1023/A:1012093209450

[36] Kasper Green Larsen and Jelani Nelson. 2017. Optimality of the Johnson-Lindenstrauss Lemma. In *FOCS*. IEEE Computer Society, 633–638. doi:10.1109/FOCS.2017.64

[37] Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. 2019. Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In *STOC*. ACM, 1027–1038. doi:10.1145/3313276.3316350 arXiv:1811.03195

[38] Ramgopal R. Mettu and C. Greg Plaxton. 2003. The Online Median Problem. *SIAM J. Comput.* 32, 3 (2003), 816–832. doi:10.1137/S0097539701383443

[39] Assaf Naor. 2018. Metric dimension reduction: A snapshot of the Ribe program. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*. 759–837. doi:10.1142/9789813272880_0029

[40] Assaf Naor and Ofer Neiman. 2012. Assouad's theorem with dimension independent of the snowflaking. *Rev. Mat. Iberoam.* 28, 4 (2012), 1123–1142. doi:10.4171/RMI/706

[41] Shyam Narayanan, Sandeep Silwal, Piotr Indyk, and Or Zamir. 2021. Randomized Dimensionality Reduction for Facility Location and Single-Linkage Clustering. In *ICML (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 7948–7957. https://proceedings.mlr.press/v139/narayanan21b.html

[42] Ofer Neiman. 2016. Low Dimensional Embeddings of Doubling Metrics. *Theory Comput. Syst.* 58, 1 (2016), 133–152. doi:10.1007/S00224-014-9567-3

[43] Kunal Talwar. 2004. Bypassing the embedding: algorithms for low dimensional metrics. In *STOC*. ACM, 281–290. doi:10.1145/1007352.1007399

[44] Luca Trevisan. 2000. When Hamming Meets Euclid: The Approximability of Geometric TSP and Steiner Tree. *SIAM J. Comput.* 30, 2 (2000), 475–485. doi:10.1137/S0097539799352735

[45] Ryan Williams. 2018. On the Difference Between Closest, Furthest, and Orthogonal Pairs: Nearly-Linear vs Barely-Subquadratic Complexity. In *SODA*. SIAM, 1207–1215. doi:10.1137/1.9781611975031.78

Received 2024-11-02; accepted 2025-02-01