Approximation Scheme for Weighted Metric Clustering via Sherali-Adams

Dmitrii Avdiukhin*
dmitrii.avdiukhin@northwestern.edu

Vaggos Chatziafratis†
vaggos@ucsc.edu

Konstantin Makarychev*
konstantin@northwestern.edu

Grigory Yaroslavtsev‡
grigory@gmu.edu

Abstract

Motivated by applications to classification problems on metric data, we study *Weighted Metric Clustering* problem: given a metric d over n points, the goal is to find a k-partition of these points into clusters $C_1, \dots, C_k$, while *minimizing* $\sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{u \in C_i} \sum_{v \in C_j} A_{ij}\, d_{uv}$, where $\mathbf{A}$ is a $k \times k$ symmetric matrix with non-negative entries. Specific choices of A lead to Weighted Metric Clustering capturing well-studied graph partitioning problems in metric spaces, such as Min-Uncut, Min-k-Sum, Min-k-Cut, and more.

Our main result is that Weighted Metric Clustering admits a polynomial-time approximation scheme (PTAS). Our algorithm handles all the above problems using the Sherali-Adams linear programming relaxation. This subsumes several prior works, unifies many of the techniques for various metric clustering objectives, and yields a PTAS for several new problems, including metric clustering on manifolds and a new family of hierarchical clustering objectives. Our experiments on the hierarchical clustering objective show that it better captures the ground-truth structural information compared to the popular Dasgupta's objective.

1 Introduction

We introduce and study Weighted Metric Clustering problem: given n points from an arbitrary metric space (V, d), we want to find a k-partition of V, i.e. a partition into k clusters $C_1, \dots, C_k$, where k is assumed to be a fixed constant. Because the quality of clustering may depend on the application at hand, we allow for a user-defined $k \times k$ symmetric matrix $\mathbf{A}$ with non-negative entries to be part of the input. Matrix $\mathbf{A}$ determines the "cost penalty" for how the k different clusters interact: if u is assigned to cluster C_i and v is assigned to cluster C_j, then the pair (u, v) pays $A_{ij} d_{uv}$, where the distance between elements u, v is denoted as d_{uv}. Hence, our goal is to *minimize* the following objective:

$$\textsc{cost}(C_1, \dots, C_k) = \sum_{i=1}^{k} \sum_{j=1}^{k} A_{ij} \mathrm{W}(C_i, C_j), \quad \text{where} \quad \mathrm{W}(C_i, C_j) = \sum_{u \in C_i} \sum_{v \in C_j} d_{uv} \qquad (\star)$$

In Weighted Metric Clustering, n is the number of input variables and k is assumed to be a fixed constant independent of n. Observe that $\mathrm{W}(C_i, C_j)$ can be thought of as an overall measure of

*Northwestern University, Illinois
†University of California Santa Cruz, California
‡George Mason University, Virginia

dissimilarity between clusters C_i and C_j (or within cluster C_i when $i = j$), which is weighted with A_{ij} in the objective ($\star$).

Note that we can interpret our objective ($\star$) as a *minimization* valued Constraint Satisfaction Problem (MIN-CSP) on variables in V and domain $\mathcal{D} = \{1, \dots, k\}$. In this CSP, we have a constraint for all pairs of variables. The weight of the constraint between variables u and v equals the distance d_{uv}. The payoff function for each constraint is defined by matrix $\mathbf{A}$: namely, the cost of assigning labels i and j to variables u and v equals A_{ij}. The goal is to find an assignment, i.e. a mapping $\ell \colon V \to \mathcal{D}$ minimizing the total payoff: $\sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{u \in V} \sum_{v \in V} A_{ij} d_{uv} \cdot \mathbb{1}\{\ell(u) = i; \ell(v) = j\}$.

The strength of objective ($\star$) lies in the flexibility of choice of matrix $\mathbf{A}$, allowing it to cover many important problems.

Metric Min-Uncut [Indyk, 1999] This is the complement of Max-Cut where we want to split into two clusters so as to minimize the sum of pairwise distances within clusters. If in ($\star$) we set $k = 2$, $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, then we pay d_{uv} only for elements u, v that end up in the same cluster.

Metric Min-k-Sum [Bartal, Charikar, and Raz, 2001] Also termed Min-k-Uncut, this is the natural extension of the previous problem to k clusters, where we want to minimize the sum of distances between pairs of points assigned to the *same* cluster. Fixing $\mathbf{A} = I_{k \times k}$ to be the $k \times k$ identity matrix yields the problem.

Metric Multiway Cut [Dahlhaus, Johnson, Papadimitriou, Seymour, and Yannakakis, 1994] We can also model problems where the cost is based on the *separated* u, v pairs. For example, taking $\mathbf{A} = J_{k \times k} - I_{k \times k}$, where J is the all-ones matrix, yields the Min-k-Cut objective, with the goal of minimizing the sum of distances among all pairs of separated points. Min-k-Cut problem additionally requires that all clusters are non-empty, and one possible approach is to fix one point per cluster; this variant of the problem, known as a *multiway cut*, is MAX SNP-hard [Dahlhaus et al., 1994] even for $k = 3$. Our algorithms are robust to such modifications of the objective and provide a PTAS for the metric case for fixed k.

A related problem is a *multicut* problem (see e.g. Costa, Létocart, and Roupin [2005]), where, given a set of k pairs $\{(s_i, t_i)\}_{i=1}^{k}$, we need to remove the edges with the smallest possible weight so that s_i and t_i are disconnected for all i. For fixed k, similarly to the multiway cut problem, we can guess clusters for all s_i and t_i.

Metric Clustering on Manifolds Our formulation can also capture problems where data points reside on a manifold. In this case, the clusters are related (they can form a chain, a ring, or a grid) and we would like to find a clustering by grouping adjacent data points. As an example, the chain topology on four clusters, i.e. $C_1 - C_2 - C_3 - C_4$, can be represent by matrix

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix},$$

indicating that pairs of points in the same cluster pay 2, pairs in the neighboring clusters pay 1, and pairs in non-neighboring clusters pay 0. Understanding such problems on manifolds served as motivation for the original work by Song, Smola, Gretton, and Borgwardt [2007] that introduced

the maximization variant of a special case of ($\star$) called Kernel Clustering. For such problems, to the best of our knowledge, no approximation was known for the minimization versions and our results provide the first PTAS.

New Application to Metric Hierarchical Clustering To highlight the versatility of our objective ($\star$), we present an application to hierarchical clustering motivated by graph compression and graph reordering problems in social networks [Dhulipala, Kabiljo, Karrer, Ottaviano, Pupyrev, and Shalita, 2016]. We introduce a novel family of minimization objectives over hierarchies which depend on the *depth* of the Lowest Common Ancestors (LCA) for pairs of leaves. In contrast, almost all prior works considered hierarchical clustering objectives based on the *size* of the LCA [Dasgupta, 2016].

2 Previous Work and Our Results

While there were important works on obtaining PTAS's for minimization problems [Indyk, 1999, de la Vega, Karpinski, and Kenyon, 2004, de la Vega, Karpinski, Kenyon, and Rabani, 2003], it was not a priori clear whether a PTAS for these problems could exist. This is mainly due to pessimistic hardness results that hold for related minimization problems: for example, for every $k > 2$ and $\varepsilon > 0$, the Min-k-Sum problem cannot be approximated within $n^{2-\varepsilon}$, even for dense graphs [Kann, Khanna, Lagergren, and Panconesi, 1996]. For more background on maximization and Min-CSPs (Appendix A).

Surprisingly, we show that every problem within our Weighted Metric Clustering ($\star$) framework admits a PTAS. As a consequence, this gives alternative PTAS for various problems, e.g., it subsumes known PTAS results for Metric Min-Uncut [Indyk, 1999] and Metric Min-k-Sum [Bartal, Charikar, and Raz, 2001]. Furthermore, we give new PTAS's for various other problems, since any matrix $\mathbf{A}$ gives rise to a new clustering problem. In particular, our framework gives the first PTAS for metric minimization version of clustering on manifolds mentioned above [Song et al., 2007], multiway cut [Dahlhaus et al., 1994], and multicut [Costa et al., 2005] problems. Furthermore, we give PTAS for a new family of hierarchical clustering objectives motivated by graph compression and graph relabeling.

An interesting aspect of our result is that a *single* algorithmic technique based on the Sherali–Adams LP relaxation can accommodate all problems. Notice that just Min-k-Sum required a variety of tools (and often ad hoc ideas) to get a PTAS: for example, the PTAS of Indyk [1999] for $k = 2$ relied on the already known PTAS for metric Max-Cut, the first non-trivial approximation of Min-k-Sum (for general k) relied on metric embeddings into hierarchically separated trees combined with dynamic programming, and finally, the PTAS of de la Vega, Karpinski, Kenyon, and Rabani [2003] used sampling and exhaustive search combined with careful reassignment of nodes to the k clusters. Our main result can be seen as a unified method that provides PTAS not only for Min-k-Sum, but all other metric problems in our framework.

Sherali–Adams. The Sherali-Adams lift-and-project method [Sherali and Adams, 1990] is a powerful technique for strengthening linear programming relaxations. This as well as other lift-and-project methods (e.g., by Lovász and Schrijver [1991]) have been extensively studied in Computer Science and Operations Research.[1] They asked if Sherali-Adams can be used to improve approximation guarantees for constraint satisfaction and combinatorial optimization problems. It turns out, that in many cases, the answer to this question is negative. Yannakakis [1988] proved

[1] See the survey by Chlamtac and Tulsiani [2012] for an overview of results.

the Traveling Salesman Problem (TSP) cannot be solved exactly using a symmetric "extended formulation" of polynomial size and, in particular, by a Sherali-Adams relaxation of polynomial size. De la Vega and Kenyon-Mathieu [2007] and Charikar, Makarychev, and Makarychev [2009a] showed that Sherali-Adams relaxation can not be used to improve approximation guarantees for many constraint satisfaction problems if we do not make additional assumptions about the structure of the CSP instances (see also Alekhnovich, Arora, and Tourlakis [2011]).

However, in some cases, Sherali-Adams can be used to obtain better approximations for Max-CSPs. In particular, Yoshida and Zhou [2014] gave a PTAS for dense instances of Max-CSPs (but not Min-CSPs!). For additional examples of Max-CSP approximations using Sherali-Adams, we refer the reader to recent papers by Thapper and Zivny [2017], Hopkins, Schramm, and Trevisan [2020], Romero, Wrochna, and Živnỳ [2021], Cohen-Addad, Lee, and Newman [2022b], Mezei, Wrochna, and Živnỳ [2023].

Kernel Clustering Motivated by applications in machine learning and statistics, Kernel Clustering was proposed by Song, Smola, Gretton, and Borgwardt [2007] as a broad family of clustering methods based on the *maximization* of dependence between the input variables and their cluster labels. It is a unified framework for various clustering methods arising from geometric, spectral or statistical considerations, and it has connections to k-means, clustering under topological constraints, and hierarchical clustering. Formally, their goal is to *maximize* objective ($\star$) under the assumption that both the distance matrix d and the cost matrix $\mathbf{A}$ are positive semidefinite. On the other hand, while we require d to be a metric, we don't require d and $\mathbf{A}$ to be positive semidefinite.

Kernel Clustering is a generalization of the positive semidefinite Grothendieck problem [Nesterov, 1998] that has found many algorithmic applications [Alon and Naor, 2004, Charikar and Wirth, 2004, Charikar, Makarychev, and Makarychev, 2009b], and has further connections to semidefinite programming, non-convex optimization and the Unique Games Conjecture [Khot and Naor, 2008, 2013]. Khot and Naor [2008, 2013] studied Kernel Clustering, presenting constant factor approximations and hardness results. In our paper, we show a PTAS for the minimization version of the problem under metric assumption.

2.1 Main Result

The main question we address here is the following:

What is the best approximation for the Weighted Metric Clustering objective ($\star$)*?*

Our main result shows that we can get an arbitrary good approximation.

Theorem 2.1. *(Informal) There is a PTAS*[2] *for the Weighted Metric Clustering objective* ($\star$)*.*

As a corollary, we get a PTAS not only for all the above-mentioned problems, but also many more, since any choice of the matrix $\mathbf{A}$ generates a new, different clustering objective. In particular, with careful choice of $\mathbf{A}$, we provide PTAS's for problems where the PTAS's were not previously known, such as clustering on manifolds and a family of hierarchical clustering objectives (Section 3), where each pair of elements is penalized depending on the depth of their least common ancestors. We describe the depth-based hierarchical clustering objectives in Section 3, with additional motivation based on the Minimum Logarithmic Arrangement presented in Appendix F, and we empirically demonstrate the advantage of these objectives in Section 6.

[2]For a minimization problem, a PTAS is an algorithm that, given $\varepsilon > 0$ as a parameter, returns a $(1+\varepsilon)$-approximation to the optimal value and runs in polynomial time for any constant ε. For maximization, we seek a $(1-\varepsilon)$-approximation.

Note that without metric assumption, we cannot have a PTAS even when $k = 3$ [Khot and Naor, 2013] under the Unique Games Conjecture, hence it's remarkable that a PTAS for the metric minimization version is possible. Moreover, we handle Weighted Metric Clustering using a single algorithmic technique via the Sherali–Adams linear programming [Sherali and Adams, 1990]. This subsumes several prior works, unifies many of the techniques on various clustering objectives, and yields PTAS's for new problems, including a new family of hierarchical clustering objectives.

Our Techniques While it is already known that the Sherali–Adams hierarchy can be used to get PTAS's for CSPs, the naïve approach would result in additive error terms, which can be acceptable for *maximization* objectives but are intolerable for *minimization* objectives, such as ($\star$). Our algorithm makes Sherali–Adams relaxations applicable to a wide class of minimization objectives and has two stages:

- Stage I assigns most of the elements via independent rounding;
- Stage II carefully handles the rest of the points, which we refer to as outliers.

To handle the outliers, we rely on the second objective $\mathrm{LP}_{\mathrm{II}}$, which is optimized simultaneously with the Sherali–Adams relaxation LP_{I}; formally, we minimize $\max(\mathrm{LP}_{\mathrm{I}}, \mathrm{LP}_{\mathrm{II}})$ and ensure that it is upper-bounded by OPT. On the other hand, a solution to $\mathrm{LP}_{\mathrm{II}}$ simplifies the process of assigning the outliers to the clusters. See Section 4 for the details.

Practical Algorithm and Experiments In Section 6, we introduce a practical version of our algorithm based on $\mathrm{LP}_{\mathrm{II}}$, which provides a constant-factor approximation to objective ($\star$). We run our experiments on 10^4 data points and show that our hierarchical clustering objective recovers a ground-truth clustering better compared to the popular Dasgupta's objective [Dasgupta, 2016].

3 Application to Hierarchical Clustering

We showcase how our general Weighted Metric Clustering framework ($\star$) can be applied to the problem of finding a hierarchy over clusters rather than a partition. In *Hierarchical Clustering* (*HC*), given a set of points V, the goal is to bijectively map the points on the leaves of a tree $\mathcal{T}$. HC is a very popular method with a wide range of applications [Leskovec, Rajaraman, and Ullman, 2020]. Recent literature [Dasgupta, 2016, Moseley and Wang, 2017, Cohen-Addad, Kanade, Mallmann-Trenn, and Mathieu, 2019] introduces a number of HC objectives where, for the hierarchical tree $\mathcal{T}$, each pair of elements (u, v) is penalized based on the number of leaves under the Lowest Common Ancestor (LCA) of u and v in $\mathcal{T}$, denoted as $\mathrm{LCA}_{\mathcal{T}}(u, v)$ (for literature review, see Appendix F). Instead of using the number of leaves under the LCA, here we propose an optimization objective for HC where the penalty term is defined based on the depth of the LCA. For a node $v \in \mathcal{T}$, let $h(v)$ denote the depth of v in the tree, defined as the number of edges on the shortest path from the root to v (e.g. $h(r) = 0$ if r is the root node). Our goal is to minimize the following over all possible binary trees $\mathcal{T}$:

$$H(\mathcal{T}) = \sum_{u,v \in V} d_{uv}\, h(\mathrm{LCA}_{\mathcal{T}}(u, v)) \qquad \text{(Depth-HC)}$$

Here d is a metric, and we shall note that HC has been extensively studied for metric spaces [Agarwala, Bafna, Farach, Paterson, and Thorup, 1998, Ailon and Charikar, 2005, Dasgupta and Long, 2005]. Objective Depth-HC captures the fact that it is better to separate the distant points early in the hierarchical structure, i.e. $h(\mathrm{LCA}_{\mathcal{T}}(u, v))$ should be small when d_{uv} is large. For HC, we show the following result (proof in Appendix F).

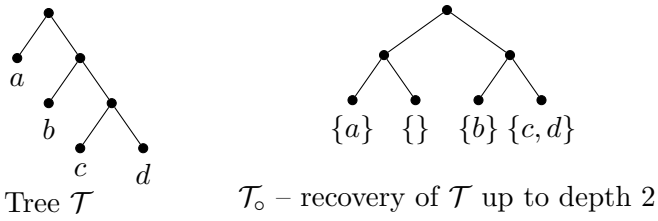

Figure 1: Recovering a tree up to a certain depth

Theorem 3.1. *For any metric d, there exists a PTAS for minimizing the objective Depth-HC.*

In order to map the HC objective to Weighted Metric Clustering ($\star$), we must appropriately choose matrix $\mathbf{A}$. The main idea is to show that it suffices to recover the tree up to the depth $\log(1/\varepsilon)$ (see Figure 1) and build random trees on deeper levels. Let $\mathcal{T}_\circ$ be a full binary tree $\mathcal{T}_\circ$ of depth $\log(1/\varepsilon)$, and we associate the $k = 1/\varepsilon$ leaves $\ell_1, \ldots, \ell_k$ of $\mathcal{T}_\circ$ with corresponding clusters $C_1, \ldots, C_k$. For different clusters C_i and C_j, we define A_{ij} as the depth of their LCA, i.e. $h(\mathrm{LCA}_{\mathcal{T}_\circ}(\ell_i, \ell_j))$. Note that if u is assigned to C_i and v is assigned to C_j, then the pair (u, v) pays A_{ij}, regardless of further partitioning of C_i and C_j. For a pair of points in the same cluster C_i, since a random binary tree splits every edge with probability $1/2$ at every level, the expected depth of the LCA is $h(\ell_i) + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \leq h(\ell_i) + 1$, which we select as A_{ii}.

Depth-based objectives are useful in Graph Compression and Vertex Reordering problems [Raghavan and Garcia-Molina, 2003, Boldi and Vigna, 2004, Chierichetti et al., 2009, Dhulipala et al., 2016], where the goal is to find space-efficient labeling schemes for the nodes in the graph. Roughly speaking, the depth $h(\mathrm{LCA}_{\mathcal{T}}(i, j))$ corresponds to the bits needed to represent a vertex in the graph, and, exploiting the fact that similar nodes tend to have similar sets of neighbors, one can significantly reduce the bit-complexity of the graph representation. A more in-depth discussion and the proof of Theorem 3.1 are deferred to Appendix F.

Extensions. We note that our result for objective Depth-HC in Theorem 3.1 also holds for more general cost functions than the hierarchical clustering objective Depth-HC specified above. For example, instead of the depth of the lowest-common ancestor, $h(\mathrm{LCA}_{\mathcal{T}}(i, j))$, we could also penalize according to the logarithm of the depth, i.e. $\log h(\mathrm{LCA}_{\mathcal{T}}(i, j))$, or the square of the depth, i.e. $h^2(\mathrm{LCA}_{\mathcal{T}}(i, j))$; our algorithms and proofs would still guarantee a PTAS in these cases. In fact, any function which depends on the depth subexponentially works. For the formal statement regarding the more general hierarchical clustering objectives, see Appendix F.

4 Sherali–Adams and Local Probability Distributions

Our $(1 + \varepsilon)$-approximation algorithm for Weighted Metric Clustering uses a Sherali–Adams relaxation for the problem. Sherali–Adams [Sherali and Adams, 1990] is a lift-and-project method for strengthening linear programming (LP) relaxations. In this paper, we will use a "local probability distribution" approach to Sherali–Adams [de la Vega and Kenyon-Mathieu, 2007, Charikar, Makarychev, and Makarychev, 2009a]. We also use a method for removing dependencies between random variables in local distributions, which was developed by Raghavendra and Tan [2012] (see also Barak, Raghavendra, and Steurer [2011] and Yoshida and Zhou [2014]).

We now describe the Sherali–Adams LP relaxation. For every tuple of points $\mathbf{v} \in V^r$, where $r \geq 2$ is a fixed integer parameter, we have a set of LP variables that defines a probability distribution of "labels" on $v_1, \ldots, v_r$. For every $\boldsymbol{\ell} \in \{1, \ldots, k\}^r$, we introduce a variable $\mathbb{P}_{\mathbf{v}}[v_1 \in$

$C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\big]$. Each of these k^r variables (sometimes called pseudo-probabilities) lies in $[0,1]$ and represents the probability that point v_i is assigned to cluster C_{ℓ_i} for all i.[3] For every $\mathbf{v} \in V^k$, the linear programming relaxation has the constraint $\sum_{\boldsymbol{\ell} \in \{1,\ldots,k\}^r} \mathbb{P}_{\mathbf{v}}\big[v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\big] = 1$. This constraint ensures that in a feasible LP solution, every $\mathbb{P}_{\mathbf{v}}$ indeed defines a local probability distribution on points $v_1, \ldots, v_r$.

We also add a constraint that guarantees that this probability does not depend on the order of points $v_1, \ldots, v_r$. For example, for $r = 2$, we impose constraint $\mathbb{P}[a \in C_1, b \in C_2] = \mathbb{P}[b \in C_2, a \in C_1]$, where a and b are arbitrary points from V. Specifically, for every permutation σ of $\{1, \ldots, k\}$, we have:

$$\mathbb{P}_{\mathbf{v}}\big[v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\big] = \mathbb{P}_{\mathbf{v}}\big[v_{\sigma(1)} \in C_{\ell_{\sigma_1}}, \ldots, v_{\sigma_k} \in C_{\ell_{\sigma_k}}\big].$$

LP variables $\mathbb{P}_{\mathbf{v}}\big[v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\big]$ prescribe probabilities to elementary events $\big\{v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\big\}$ and thus define probabilities for all events: for $\mathcal{E} \subseteq \{1, \ldots, k\}^r$, we let $\mathbb{P}_{\mathbf{v}}[\mathbf{v} \in \mathcal{E}] = \sum_{\boldsymbol{\ell} \in \mathcal{E}} \mathbb{P}_{\mathbf{v}}\big[v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\big]$. In other words, $\mathbb{P}_{\mathbf{v}}[\mathbf{v} \in \mathcal{E}]$ is the probability that labels for $v_1, \ldots, v_r$ drawn from local distribution $\mathbb{P}$ are $C_{\ell_1}, \ldots, C_{\ell_r}$ (respectively) with $\boldsymbol{\ell} \in \mathcal{E}$. To avoid ambiguity, we will use a different notation to denote probabilities associated with our algorithm. We shall write $\Pr[v_1 \in X_1, \ldots, v_r \in X_r]$ to denote the probability that points $v_1, \ldots, v_r$ belong to random sets $X_1, \ldots, X_r$ chosen by the algorithm.

An important constraint of the Sherali–Adams relaxation is that *all local distributions are locally consistent*, as we explain next. Consider two tuples $\mathbf{u}$ and $\mathbf{v}$. Let $\mathbf{z}$ be the set of common points in $\mathbf{u}$ and $\mathbf{v}$. Both $\mathbf{u}$ and $\mathbf{v}$ define marginal probability distributions on cluster labels for points in $\mathbf{z}$. We require that these marginal distributions be the same. Specifically, we add a constraint to the linear program that enforces that label distributions on $\mathbf{u}$ and $\mathbf{v}$ agree on the intersection $\mathbf{z} = \mathbf{u} \cap \mathbf{v}$. We denote the marginal probability distribution on every set $\mathbf{z}$ of size at most r by $\mathbb{P}_{\mathbf{z}}$. If $\mathbf{z}$ consists of one point u or two points u,v, we write $\mathbb{P}_u$ and $\mathbb{P}_{uv}$, respectively.

We stress that even though all local distributions $\mathbb{P}$ are locally consistent, generally speaking, there is no global distribution of cluster labels that is consistent with all local distributions. We also note that the size of the Sherali–Adams is exponential in r, since the number of variables equals $n^r \cdot k^r$. Thus, if we want to solve a Sherali–Adams relaxation in polynomial time, the parameter r must be a constant.

When each variable in a solution to the Sherali–Adams relaxation is equal to 0 or 1, we call the solution integral. An integral solution corresponds to an actual clustering in which u belongs to C_i if and only if $\mathbb{P}_u[u \in C_i] = 1$. Moreover, $\mathbb{P}_{\mathbf{v}}\big[v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\big] = 1$ if and only if $v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}$. That is, $\mathbb{P}_{\mathbf{v}}\left[v_1 \in C_{\ell_1}, \ldots, v_r \in C_{\ell_r}\right] = \mathbb{1}\{v_1 \in C_{\ell_r}, \ldots, v_r \in C_{\ell_r}\}$, where $\mathbb{1}\{\mathcal{E}\}$ is the indicator of the event $\mathcal{E}$. We now define the objective function for our Sherali–Adams relaxation and introduce some additional constraints. We assume that we know the sizes of the optimal clusters $n_1 = |C_1^*|, \ldots, n_k = |C_k^*|$. We additionally assume that we know their centers $c_1 \in C_1^*, \ldots, c_k \in C_k^*$ which guarantee 3-approximation (see Lemma B.2). Note that there are at most $O(n^{2k})$ combinations of different c_i's and n_j's, and hence we can try all possibilities. We use $\boldsymbol{\Pi}$ to denote the particular choice of c_i's and n_j's and call it the clustering profile.

The objective of our linear programming relaxation is the maximum of LP_{I} and $\mathrm{LP}_{\mathrm{II}}$ under the constraints above:

$$\textbf{minimize } LP = \max(\mathrm{LP}_{\mathrm{I}}, \mathrm{LP}_{\mathrm{II}}), \tag{1}$$

[3] Formally, one should think about assigning point v_i to C_{ℓ_i} as of assigning label ℓ_i to point v_i

$$\mathrm{LP_I} = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\sum_{u,v\in V} A_{ij}\, d_{uv}\, \mathbb{P}_{uv}\big[u \in C_i,\, v \in C_j\big]$$

$$\mathrm{LP_{II}} = \frac{1}{3}\sum_{i=1}^{k}\sum_{u\in V} F_{\mathbf{\Pi}}(u,i)\, \mathbb{P}_u\big[u \in C_i\big],$$

$$F_{\mathbf{\Pi}}(u,i) = \sum_{j=1}^{k} n_j\, a_{ij}\, d_{uc_{i\wedge j}}, \tag{2}$$

where $i \wedge j$ is defined as $\min(i,j)$. The first objective $\mathrm{LP_I}$ is a direct relaxation of the objective function of Weighted Metric Clustering: in an integral LP solution – when each $\mathbb{P}_{uv}\big[u \in C_i,\, v \in C_j\big]$ is 0 or 1 – the value of $\mathrm{LP_I}$ equals the cost of the corresponding combinatorial solution to Weighted Metric Clustering. Consequently, in the optimal integral solution to the problem, $\mathrm{LP_I} = \mathrm{OPT}$, where $\mathrm{OPT} = \textsc{cost}(C_1^*, \ldots, C_k^*)$ is the value of the optimal solution.

The second objective $\mathrm{LP_{II}}$ is upper bounded by OPT in the optimal integral solution by Lemma B.2 when c_i's and n_j's are guessed correctly. This is due to the fact that for any $u \in V$, $i \in [k]$, and the correct guess of n_i and c_i, $n_i\, d_{uc_i}$ is a good approximation of $\sum_{v\in C_j} d_{uv}$ [de la Vega, Karpinski, Kenyon, and Rabani, 2003]. Therefore, $\mathrm{OPT}_{\mathrm{LP_I}} \le \mathrm{OPT}$ and $\mathrm{OPT}_{\mathrm{LP_{II}}} \le \mathrm{OPT}$, where $\mathrm{OPT}_{\mathrm{LP_I}}$ and $\mathrm{OPT}_{\mathrm{LP_{II}}}$ are the values of $\mathrm{LP_I}$ and $\mathrm{LP_{II}}$ in the optimal solution to our linear program.

Intuitively, $\mathrm{LP_{II}}$ is used for bounding the error terms in the analysis and, compared to $\mathrm{LP_{II}}$, has the following advantages. First, every term involves a single point u (note that other variables in each term are either guessed or fixed), and hence it's easy to optimize. Second, $\mathrm{LP_{II}}$ refers to cluster centers instead of clusters themselves, which is important for the case when there are multiple equivalent solutions to the original problem, e.g. in the case of Min-Uncut. In the analysis, we often use triangle inequality to bound $d_{uv} \le d_{uc} + d_{cv}$, with the choice of c being crucial. $\mathrm{LP_{II}}$ forces the center for each cluster, which makes the choice of c clear in each particular case.

Finally, we add capacity constraints to our relaxation, which are satisfied in the integral solution to Weighted Metric Clustering. For all $i \in \{1, \ldots, k\}$: $\sum_{u\in V} \mathbb{P}_u[u \in C_i] \le n_i$. These constraints are important since $\mathrm{LP_{II}}$ is a good approximation of ($\star$) only if cardinalities are guessed and enforced correctly.

4.1 Making Point Distributions Nearly Independent

We now define *nearly independent* local distributions and then describe a procedure MakeIndependent that transforms local distributions $\mathbb{P}$ obtained by solving the Sherali–Adams LP relaxation into a nearly independent local distributions $\mathbb{P}^*$. This procedure uses the *conditional probability technique* for Sherali–Adams [Raghavendra and Tan, 2012]. The main difference between our result and theirs is that we require that local distributions $\mathbb{P}^*$ (see below) are simultaneously nearly independent for k sets $D_1, \ldots, D_k$, while Raghavendra and Tan [2012] obtain a globally uncorrelated solution which corresponds to the case when we have only one set $A = V$. For us, it is crucial to have sets $D_1, \ldots, D_k$ in the definition because some sets D_i may have size $o(n)$ (e.g., $\sqrt{n}$). In that case, the guarantees of the algorithm by Raghavendra and Tan [2012] are not sufficient for us.

First, we introduce some notation. Denote the distribution of pairs u and v in which u and v are sampled independently with distributions $\mathbb{P}_u$ and $\mathbb{P}_v$ by $\mathbb{P}_u \otimes \mathbb{P}_v$:

$$(\mathbb{P}_u \otimes \mathbb{P}_v)[u \in C_i,\, v \in C_j] = \mathbb{P}_u[u \in C_i] \cdot \mathbb{P}_v[v \in C_j].$$

Definition 4.1. Let $D_1, \dots, D_k$ be subsets of V. We say that a family of local probability distributions $\{\mathbb{P}\}$ are (γ, δ)-nearly independent for sets $D_1, \dots, D_k$ if the following condition holds: for every $u \in V$ and every $j \in \{1, \dots, k\}$, for all but γ fraction of v in D_j, we have $\|\mathbb{P}_u \otimes \mathbb{P}_v - \mathbb{P}_{u,v}\|_{TV} \le \delta$. Equivalently, for all $u \in V$ and $i \in [k]$, the number of elements $v \in D_j$ such that $\|\mathbb{P}_u \otimes \mathbb{P}_v - \mathbb{P}_{u,v}\|_{TV} > \delta$ must be at most $\gamma |D_i|$. If $\|\mathbb{P}_u \otimes \mathbb{P}_v - \mathbb{P}_{u,v}\|_{TV} \le \delta$, we say that u and v are δ-nearly independent according to $\mathbb{P}_{u,v}$.

Theorem 4.2. *For every $\delta, \gamma, \eta \in (0, 1)$ and integer $k > 1$, there exists a randomized polynomial-time procedure that given a solution $\mathbb{P}$ to the Sherali–Adams relaxation with $r \ge 2 + \frac{k \log_2 k}{2\delta^2 \gamma \eta}$ rounds, outputs a family of local probability distributions $\{\mathbb{P}^*_u\}_u$ and $\{\mathbb{P}^*_{uv}\}_{uv}$ and exit status ("success" or "failure") such that*

1. *If the algorithm succeeds, then $\mathbb{P}^*$ is (γ, δ)-nearly independent.*
2. *For all $u, v \in V$ and $i, j \in \{1, \dots, k\}$,*

$$\mathbb{E}\left[\mathbb{P}^*_u[u \in C_i]\right] = \mathbb{P}_u[u \in C_i]$$
$$\mathbb{E}\left[\mathbb{P}^*_{uv}[u \in C_i, v \in C_j]\right] = \mathbb{P}_{uv}[u \in C_i, v \in C_j].$$

3. *The algorithm fails with probability at most η.*

The goal of algorithm MakeIndependent is to build a (γ, δ)-nearly independent family $\{\mathbb{P}^*\}$ while preserving the expectation of the LP value. The algorithm builds a sequence of distributions $\{\mathbb{P}^{(0)}\} = \{\mathbb{P}\}, \{\mathbb{P}^{(1)}\}, \dots$. At iteration t, it finds a point u violating the (γ, δ)-nearly independence condition, and then *conditions* local distributions on the event $\mathbb{P}^{(t)}[u \in C_i]$ for i drawn from distribution $\mathbb{P}^{(t)}_u$. Loosely speaking, every time we do the conditioning step, we make more pairs (u, v) nearly independent. We show that a certain measure – entropy – decreases with each iteration by at least a fixed amount, and hence in approximately r steps, we get nearly independence with the desired parameters. We provide more details and prove this theorem in Appendix C.

5 Main Algorithm

In this section, we outline our $(1 + \varepsilon)$-approximation algorithm or PTAS (polynomial-time approximation scheme) for the Weighted Metric Clustering problem. We provide full details in Appendices D and E. The pseudocode is provided in Algorithm 1.

Algorithm Outline In the first step, the algorithm guesses the cluster centers $\{c_i\}$ and sizes $\{n_j\}$, which we call the clustering profile and denote by $\mathbf{\Pi}$. Note that all choices of $\{c_i\}$ and $\{n_j\}$ can be enumerated in polynomial time, and our analysis assumes the correct choice. Then, the algorithm solves the r-round Sherali–Adams relaxation for Weighted Metric Clustering (see Section 4) and obtains local distributions $\mathbb{P}$. For constant r, the size of the relaxation is polynomial in n, and thus it can be solved in polynomial time. We then assign points to clusters using a two-stage algorithm.

At Stage I, we assign most points to clusters $X_1, \dots, X_k$ and place the remaining points, which we call "outliers", in set O. We guarantee that the cost of the partial clustering $X_1, \dots, X_k$ is at most $(1 + \varepsilon)$OPT in expectation and each point is an outlier with probability at most ηk (see Lemma D.2 for the formal statement), where η is a small parameter depending on ε.

Algorithm 1: PTAS for Weighted Metric Clustering

input : V – set of points, $\{d_{uv}\}_{u,v\in V}$ – pairwise distances, $\{A_{ij}\}_{i,j=1}^{k}$ – inter-cluster costs

parameters: r – number of rounds of SA relaxation, η – outlier probability threshold, δ – fraction of dependent points, γ – independence threshold

Guess cluster centers $c_1, \dots, c_k$ and sizes $n_1, \dots, n_k$

Let $\{\mathbb{P}\}$ be the r-round solution to SA relaxation for Problem (1)

$D_i = \{u \in V : \mathbb{P}_u[u \in C_i] \geq \eta\}$ for all i

$\{\mathbb{P}^*\} = \textsc{MakeIndependent}(\{\mathbb{P}\}, \{D_i\}, \delta, \gamma)$

// Tentative assignment via independent rounding

for *all* $u \in V$ **do**

  Assign u to C_i with probability $\mathbb{P}^*[u \in C_i]$

// Stage I: Assigning non-outliers

if $\mathbb{P}^*$ *is* (γ, δ)*-nearly independent for* $D_1, \dots, D_k$ **then**

  $X_i = C_i \cap D_i, \quad O = \bigcup_i (C_i \setminus D_i)$

else

  $O = V$ // Every point is outlier

// Stage II: Assigning outliers

for *all* $u \in O$ **do**

  Assign u to Y_i with probability $\mathbb{P}[u \in C_i]$.

return $(X_1 \cup Y_1, \dots, X_k \cup Y_k)$

Algorithm 2: $\textsc{MakeIndependent}(\{\mathbb{P}\}, \{D_i\}, \delta, \gamma)$

input : $\{\mathbb{P}\}$ – r-round solution to SA relaxation, $\{D_i\}_{i=1}^{k}$ – candidate sets for each cluster, δ – fraction of dependent points, γ – independence threshold

Let $\{\mathbb{P}^{(0)}\}$ be $\{\mathbb{P}\}$

for $t = 0, 1, \dots, r-3$ **do**

  if $\{\mathbb{P}^{(t)}\}$ *is* (γ, δ)*-nearly independent for sets* $D_1, \dots, D_k$ *(Def. 4.1)* **then**

    return $\{\mathbb{P}^{(t)}\}$

  Let u be a point violating the (γ, δ)-nearly independence condition.

  Assign u to C_i with probability $\mathbb{P}^{(t)}[u \in C_i]$.

  Let $\{\mathbb{P}^{(t+1)}\}$ be $\{\mathbb{P}^{(t)}\}$ conditioned on $u \in C_i$.

return $\{\mathbb{P}^{(r-2)}\}$

At Stage II, we cluster the outliers from set O. For this purpose, we use a variant of the 3-approximation algorithm, which we provide in Section B. Since the number of outliers is very small, the cost of clustering them is also small despite the fact that we use a constant factor approximation for outliers. Finally, we combine the clusterings obtained at Stage I and Stage II and get a clustering of cost at most $(1+\varepsilon)$OPT. The algorithm for clustering outliers is discussed in Appendix E.

Stage I We now examine the first stage of the algorithm in more detail. It is inspired by Yoshida and Zhou [2014] and Raghavendra and Tan [2012]. The general idea is to transform the solution for the Sherali–Adams relaxation to a family of local distributions $\{\mathbb{P}^*\}$ such that

$$\mathbb{P}^*_{uv}[u \in C_i; v \in C_j] \approx \mathbb{P}^*_u[u \in C_i] \cdot \mathbb{P}^*_v[v \in C_j] \tag{3}$$

for most pairs of points. This can be done using the method discussed in the previous section. Next, we want to randomly and independently assign every point u to cluster i with probability $\mathbb{P}_u[u \in C_i]$. If condition (3) holds for some pair (u, v) and all i, j, then the expected cost this algorithm pays for clustering pair (u, v), $\sum_{ij} d_{uv} A_{ij} \mathbb{P}_u\big[u \in C_i\big] \cdot \mathbb{P}_v\big[v \in C_j\big]$, is approximately equal to the LP cost of this pair, $\sum_{ij} d_{uv} A_{ij} \mathbb{P}_{uv}\big[u \in C_i, v \in C_j\big]$. The problem, however, is that condition (3) does not hold for all pairs (u, v). Furthermore, it may happen that the algorithm creates a very expensive small cluster X_i such that for all pairs $u, v \in X_i$ we do not have approximate equality (3). Consequently, the cost of such a cluster cannot be charged to the LP relaxation.

The discussion above leads to the following idea: let us make local distributions not only nearly independent for most pairs (u, v) but nearly independent for each point u and most v's in each cluster the algorithm creates. This is formally stated in Definition 4.1. However, the problem is that the algorithm does not know in advance what clusters it is going to produce. So, it uses a proxy for these clusters – sets of candidate points $D_1, \ldots, D_k$. Set D_i contains points that are somewhat likely to be assigned to cluster i.

We now summarize Stage I. First, the algorithm solves the Sherali–Adams relaxation. Then, it defines sets of candidates $D_1, \ldots, D_k$, where each D_i contains points u for which $\mathbb{P}[u \in C_i] \geq \eta$ (where η is a small constant depending on ε). It calls algorithm MakeIndependent (described Section 4.1) with sets $D_1, \ldots, D_k$ and obtains (γ, δ)-nearly independent local distributions $\mathbb{P}^*$. Next, it randomly assigns points to clusters using distribution $\mathbb{P}^*$. To make sure that we can pay for each created cluster X_i, this cluster needs to be a subset of the corresponding candidate set D_i. Thus, if point u is assigned to X_i but u is not in D_i, we remove u from X_i and mark u as an outlier. Stage I returns sets $X_1, \ldots, X_k$ along with the set of outliers O, which are assigned to clusters at Stage II. We can now charge the cost of all pairs (u, v) that are nearly independent to the LP objective. Using triangle inequalities, we can also bound the cost of all other pairs (u, v) in $V \setminus O$. We provide all details in Appendix D.

Stage II At Stage II, we assign outliers to clusters. Our approach to dealing with outliers is somewhat similar to the approach introduced by Makarychev, Makarychev, and Razenshteyn [2019]. As discussed above, the number of outliers is small, which is one of the main reasons why their assignment does not significantly change the objective. The outliers are assigned using independent rounding based on $\mathbb{P}$ (instead of $\mathbb{P}^*$ for non-outliers). In Theorem E.3, we analyze the cost of assigning outliers to clusters. Putting everything together, we prove that our algorithm provides a PTAS.

Theorem 5.1. *For $\delta = \gamma = \frac{\eta^2 \varepsilon}{9}$ and $r = 2 + \frac{k \log_2 k}{2\delta^2 \gamma \eta}$, Algorithm 1 finds clustering with the expected objective value within $(1 + \varepsilon)$-factor of* OPT *with probability at least $1 - \eta$ for any $\eta \leq \frac{\varepsilon^2}{90k^2}$.*

6 Experiments

In this section, we perform experiments on the hierarchical clustering objective (Depth-HC) defined in Section 3:

$$H(\mathcal{T}) = \sum_{u,v \in V} d_{uv}\, h(\mathrm{LCA}_{\mathcal{T}}(u, v))$$

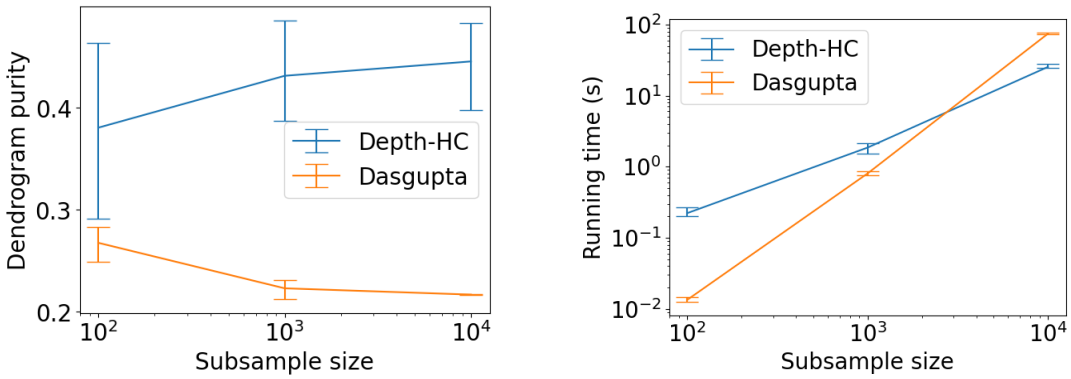

Figure 2: Comparison of the depth-based objective (Depth-HC) and the Dasgupta's objective. Data points correspond to averages over 10 runs, and error bars correspond to the 10% and 90% quantiles. The experiments are performed on a single-core Intel Xeon 2.2GHz CPU.

For our experiments, we use a simplified version of the algorithm, based on the LP_{II} relaxation from Section 4, which achieves 3-approximation (Appendix B):

$$\sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{u \in V} n_j \, A_{ij} \, d_{u c_{i \wedge j}} \, \mathbb{P}_u\big[u \in C_i\big],$$

where $n_1, \ldots, n_k$ are cluster cardinalities and $c_1, \ldots, c_k$ are cluster centers. This objective can be optimized efficiently as an instance of a minimum-cost flow problem, while precisely satisfying the imposed cardinality constraints. We run the algorithm multiple times with different guesses of $\{n_i\}$ and $\{c_i\}$, and, since the guesses might be not precise, we improve the resulting solution using local search.

Datasets We perform evaluation on various hierarchical datasets. In this section, we present experiments on random subsamples (of sizes 10^2, 10^3, and 10^4) of a well-known 20 NEWSGROUPS dataset [Lang, 1995], and in Appendix G we present additional experiments on ZEBRAFISH [Wagner et al., 2018], CIFAR-10 [Krizhevsky and Hinton, 2009], and other datasets. The inputs in 20 NEWSGROUPS are text documents, which we transform to the Euclidean vectors using a pre-trained language model (see Appendix G for details). Finally, we use the ground-truth hierarchical structure to obtain a flat clustering based on the top-level split.

Objectives We compare the following objectives:

- **Depth-based objective** (Depth-HC). Based on the algorithm from Section 3, we approximate the objective by building a hierarchical tree up to a certain level and building random trees on the resulting clusters. We select the level ℓ so that the number of clusters 2^ℓ is close to the number of ground-truth clusters.
- **Dasgupta's objective** [**Dasgupta, 2016**], defined as $\sum_{u<v} w(u,v)\,|\mathrm{LCA}_{\mathcal{T}}(u,v)|$, where w is the similarity between items. We convert distances to similarities using the standard RBF kernel: $w(x,y) = \exp\left(-\frac{\|x-y\|^2}{2}\right)$. We optimize Dasgupta's objective using recursive Min-Cut [Chatziafratis et al., 2020], for which we use METIS [Karypis and Kumar, 1995].

Evaluation and Results We evaluate how well the above objectives recover the ground-truth clustering information using the dendrogram purity objective:

$$DP(\mathcal{T}) = \frac{1}{\sum_{i=1}^{m} |C_i|^2} \sum_{i=1}^{m} \sum_{u,v \in C_i} \frac{|C_i \cap \mathrm{LCA}_{\mathcal{T}}(u,v)|}{|\mathrm{LCA}_{\mathcal{T}}(u,v)|},$$

where $C_1, \ldots, C_m$ are the ground-truth clusters. Intuitively, this objective measures how well-separated are the ground-truth clusters in the tree.

Figure 2 shows that (Depth-HC) objective achieves significantly better dendrogram purity compared with Dasgupta's objective. Moreover, the complexity of our algorithm is noticeably slower, and, with the increase in the number of data points, the gap in quality increases, exceeding the factor of two for 10^4 points. To conclude, these experiments demonstrate the usefulness of our hierarchical objective as well as the existence of efficient approaches for its optimization.

We provide additional experiments in Appendix G.

7 Conclusion

In this paper, we describe a polynomial-time approximation scheme (PTAS) for the Weighted Metric Clustering problem based on the Sherali–Adams relaxation. This in turn provides a PTAS for many important special cases, including metric clustering on manifolds and our novel depth-based hierarchical clustering objective. In our experiments, we compare our hierarchical clustering objective with Dasgupta's objective and show that our objective recovers the ground-truth clustering information more precisely.

An interesting open question would be handling cardinality constraints, which covers important applications such as variations of the sparsest cut or objectives with balance constraints. Another interesting variation of the objective is a sum of multiple objectives of the form ($\star$).

Acknowledgements

Konstantin Makarychev is supported by the NSF Awards CCF-1955351, CCF-1934931, and EECS-2216970.

References

Richa Agarwala, Vineet Bafna, Martin Farach, Mike Paterson, and Mikkel Thorup. On the approximability of numerical taxonomy (fitting distances by tree metrics). *SIAM Journal on Computing*, 28(3):1073–1085, 1998.

Nir Ailon and Noga Alon. Hardness of fully dense problems. *Information and Computation*, 205 (8):1117–1129, 2007.

Nir Ailon and Moses Charikar. Fitting tree metrics: Hierarchical clustering and phylogeny. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 73–82. IEEE, 2005.

Mikhail Alekhnovich, Sanjeev Arora, and Iannis Tourlakis. Towards strong nonapproximability results in the lovász-schrijver hierarchy. *computational complexity*, 20:615–648, 2011.

Noga Alon and Assaf Naor. Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the 36th annual ACM symposium on Theory of computing*, pages 72–80, 2004.

Noga Alon, W Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csp problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 232–239, 2002.

Noga Alon, Yossi Azar, and Danny Vainstein. Hierarchical clustering: A 0.585 revenue approximation. In *Conference on Learning Theory*, pages 153–162. PMLR, 2020.

Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of np-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999. doi: 10.1006/jcss.1998.1605. URL https://doi.org/10.1006/jcss.1998.1605.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1): 89–113, 2004.

Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 472–481. IEEE, 2011.

Yair Bartal, Moses Charikar, and Danny Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 11–20, 2001.

Cristina Bazgan, Wenceslas Fernandez de la Vega, and Marek Karpinski. Polynomial time approximation schemes for dense instances of minimum constraint satisfaction. *Random Struct. Algorithms*, 23(1):73–91, 2003. doi: 10.1002/rsa.10072. URL https://doi.org/10.1002/rsa.10072.

Paolo Boldi and Sebastiano Vigna. The webgraph framework i: compression techniques. In *Proceedings of the 13th international conference on World Wide Web*, pages 595–602, 2004.

Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–854. SIAM, 2017.

Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 54–60. IEEE, 2004.

Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Integrality gaps for sherali-adams relaxations. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 283–292, 2009a.

Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for maximum constraint satisfaction problems. *ACM Transactions on Algorithms (TALG)*, 5(3): 1–14, 2009b.

Moses Charikar, Mohammad Taghi Hajiaghayi, Howard Karloff, and Satish Rao. ℓ_2^2 spreading metrics for vertex ordering problems. *Algorithmica*, 56(4):577–604, 2010.

Vaggos Chatziafratis, Grigory Yaroslavtsev, Euiwoong Lee, Konstantin Makarychev, Sara Ahmadian, Alessandro Epasto, and Mohammad Mahdian. Bisect and Conquer: Hierarchical Clustering via Max-Uncut Bisection. In *Proceedings of the 33rd International Conference on Artificial Intelligence and Statistics*, pages 3121–3132. PMLR, June 2020.

Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 219–228, 2009.

Eden Chlamtac and Madhur Tulsiani. Convex relaxations and integrality gaps. *Handbook on semidefinite, conic and polynomial optimization*, pages 139–169, 2012.

Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM*, 66(4):1–42, 2019.

Vincent Cohen-Addad, Debarati Das, Evangelos Kipouridis, Nikos Parotsidis, and Mikkel Thorup. Fitting distances by tree metrics minimizing the total error within a constant factor. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 468–479. IEEE, 2022a.

Vincent Cohen-Addad, Euiwoong Lee, and Alantha Newman. Correlation clustering with sherali-adams. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 651–661. IEEE, 2022b.

Marie-Christine Costa, Lucas Létocart, and Frédéric Roupin. Minimal multicut and maximal integer multiflow: A survey. *European Journal of Operational Research*, 162(1):55–69, April 2005. ISSN 0377-2217. doi: 10.1016/j.ejor.2003.10.037.

E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994. doi: 10.1137/S0097539792225297. URL https://doi.org/10.1137/S0097539792225297.

Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127, 2016.

Sanjoy Dasgupta and Philip M Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005.

W Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 50–58, 2003.

Wenceslas Fernandez de la Vega and Claire Kenyon-Mathieu. Linear programming relaxations of maxcut. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 53–61, 2007.

Wenceslas Fernandez de la Vega, Marek Karpinski, and Claire Kenyon. Approximation schemes for metric bisection and partitioning. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 506–515. SIAM, 2004. URL http://dl.acm.org/citation.cfm?id=982792.982864.

Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. Compressing graphs and indexes with recursive graph bisection. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1535–1544, 2016.

W Fernandez de la Vega. Max-cut has a randomized approximation scheme in dense graphs. *Random Structures & Algorithms*, 8(3):187–198, 1996.

Alan M. Frieze and Ravi Kannan. The regularity lemma and approximation schemes for dense problems. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 12–20. IEEE Computer Society, 1996. doi: 10.1109/SFCS.1996.548459. URL https://doi.org/10.1109/SFCS.1996.548459.

Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *arXiv preprint cs/0504023*, 2005.

Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4): 798–859, 2001.

Samuel B Hopkins, Tselil Schramm, and Luca Trevisan. Subexponential lps approximate max-cut. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 943–953. IEEE, 2020.

P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. In *40th Annual Symposium on Foundations of Computer Science*, pages 154–159, October 1999. doi: 10.1109/SFFCS.1999.814587.

Viggo Kann, S. Khanna, Jens Lagergren, and A. Panconesi. On the hardness of approximating max k-cut and its dual. In *Israeli Symposium on Theoretical Computer Science*, 1996.

George Karypis and Vipin Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. *University of Minnesota*, 1995.

Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002.

Subhash Khot and Assaf Naor. Approximate kernel clustering. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 561–570. IEEE Computer Society, 2008.

Subhash Khot and Assaf Naor. Sharp kernel clustering algorithms and their associated grothendieck inequalities. *Random Structures & Algorithms*, 42(3):269–300, 2013.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Ken Lang. NewsWeeder: Learning to Filter Netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Morgan Kaufmann, San Francisco (CA), January 1995. ISBN 978-1-55860-377-6. doi: 10.1016/B978-1-55860-377-6.50048-7.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020.

L. Lovász and A. Schrijver. Cones of Matrices and Set-Functions and 0–1 Optimization. *SIAM Journal on Optimization*, 1(2):166–190, May 1991. ISSN 1052-6234. doi: 10.1137/0801013.

Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1027–1038. Association for Computing Machinery, June 2019. doi: 10.1145/3313276.3316350.

Balázs F Mezei, Marcin Wrochna, and Stanislav Živnỳ. Ptas for sparse general-valued csps. *ACM Transactions on Algorithms*, 19(2):1–31, 2023.

Benjamin Moseley and Joshua Wang. Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-means, and Local Search. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Benjamin Moseley, Sergei Vassilvtiskii, and Yuyan Wang. Hierarchical clustering in general metric spaces using approximate nearest neighbors. In *International Conference on Artificial Intelligence and Statistics*, pages 2440–2448. PMLR, 2021.

Yu Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization methods and software*, 9(1-3):141–160, 1998.

Sriram Raghavan and Hector Garcia-Molina. Representing web graphs. In *Proceedings 19th International Conference on Data Engineering*, pages 405–416. IEEE, 2003.

Prasad Raghavendra and Ning Tan. Approximating csps with global cardinality constraints using sdp hierarchies. In *Proceedings of the 33rd annual ACM-SIAM symposium on Discrete Algorithms*, pages 373–387. SIAM, 2012.

Miguel Romero, Marcin Wrochna, and Stanislav Živnỳ. Treewidth-pliability and ptas for max-csps. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 473–483. SIAM, 2021.

Hanif D Sherali and Warren P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

Le Song, Alex Smola, Arthur Gretton, and Karsten M Borgwardt. A dependence maximization view of clustering. In *Proceedings of the 24th international conference on Machine learning*, pages 815–822, 2007.

Johan Thapper and Stanislav Zivny. The power of sherali–adams relaxations for general-valued csps. *SIAM Journal on Computing*, 46(4):1241–1279, 2017.

Daniel E. Wagner, Caleb Weinreb, Zach M. Collins, James A. Briggs, Sean G. Megason, and Allon M. Klein. Single-cell mapping of gene expression landscapes and lineage in the zebrafish embryo. *Science*, 360(6392):981–987, June 2018. doi: 10.1126/science.aar4362.

Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. In *Proceedings of the 20th annual ACM symposium on Theory of computing*, pages 223–228, 1988.

Yuichi Yoshida and Yuan Zhou. Approximation schemes via sherali-adams hierarchy for dense constraint satisfaction problems and assignment problems. In *Innovations in Theoretical Computer Science, 2014*, pages 423–438. ACM, 2014. doi: 10.1145/2554797.2554836. URL https://doi.org/10.1145/2554797.2554836.

A Background on CSPs

Maximization vs. Minimization There is an important distinction between minimization and maximization clustering objectives, in that minimization versions are often much harder to approximate compared with their maximization counterparts. This is because in minimization problems the error must be compared against a potentially very small optimum value, which might even be equal to 0. On the other hand, for Max-CSPs a trivial – yet often a reasonably good solution – is a random assignment, which yields constant-factor approximations.[4] In contrast, for Min-CSPs, even obtaining polylog n approximations has been challenging. Historically, this led researchers to first try to obtain Polynomial Time Approximation Schemes (PTAS) for Max-CSPs. Unfortunately, even certain Max-CSPs like Max-Cut or Max-3-SAT cannot have PTAS's unless $P = NP$ [Khot, 2002, Håstad, 2001], leading researchers to consider restrictions to the inputs. A common approach is to try to find a PTAS under well-motivated restrictions, e.g. assuming that the input is either *dense* or comes from a *metric*. To the best of our knowledge, both of the Weighted Metric Clustering formulations mentioned above have not been studied under density or metric assumptions.

Density of CSPs Fernandez de la Vega [1996], Arora, Karger, and Karpinski [1999], Frieze and Kannan [1996] show that there exists a PTAS for Max-Cut (and various other Max-CSPs) if the graph is *dense*, i.e., it has $\Omega(n^2)$ edges. These works led to a series of approximation results for the properties of dense Max-r-CSPs instances of arity r. Roughly speaking, "dense" means that there are $\Omega(n^r)$ constraints (other slightly different notions of density have also been considered). For example, a prominent dense problem is a version of the Correlation Clustering problem [Bansal, Blum, and Chawla, 2004], where the input is assumed to be a complete graph with + or − edges. Correlation Clustering with fixed number of clusters k is yet another example where obtaining PTAS for minimization was much more challenging than maximization [Giotis and Guruswami, 2005]; in fact, if k is allowed to be part of the input, the minimization version is APX-hard (and hence unlikely to have PTAS), even though the maximization version admits a PTAS [Charikar, Guruswami, and Wirth, 2005]. For more hardness results for minimization problems, even on dense instances, see Ailon and Alon [2007].

PTAS for Max vs Min CSPs. It is important to note that due of the density, a random assignment satisfies a constant fraction of constraints, so dense Max-r-CSPs have optimum value $\Theta(n^r)$. This implies that in order to get a PTAS for dense Max-r-CSPs, we can design an algorithm with *additive* error εn^r [Alon, de la Vega, Kannan, and Karpinski, 2002]. In contrast, for minimization problems, algorithms with additive error are insufficient since the optimal value can be very small or even 0. In fact, there are many Min-CSPs, such as Min-3-Uncut, that provably do not have PTAS, even under extreme density assumptions. Somewhat surprisingly, Bazgan et al. [2003] gave a PTAS for dense minimization versions of SAT formulas like Min-r-SAT.

Unified PTAS for Max-CSPs. The aforementioned line of research led to significant algorithmic developments (smooth integer programs, random sampling and exhaustive search, connections to low-degree polynomials etc.) and culminated in a paper by Yoshida and Zhou [2014] which unified all these results by providing a single PTAS that worked for all Max-CSPs, based on the Sherali–Adams linear programming (LP) relaxation hierarchy. Their main result states that for

[4] For example, random assignment for 3-SAT obtains a $\frac{7}{8}$-approximation, but no approximation exists for the minimum 3-UNSAT problem unless $P = NP$.

any $\varepsilon > 0$, Sherali–Adams LP with roughly $\mathcal{O}\left(1/\varepsilon^2\right)$ rounds gives a $(1-\varepsilon)$-approximation to dense Max-r-CSPs.

B 3-Approximation Algorithm

In this Section, we first present a 3-approximation algorithm for Weighted Metric Clustering. This result will be then used in the analysis of the outlier assignment step in Algorithm 1, namely for bounding the error terms. In the following sections, we use notation $\mathrm{W}(C_i, C_j) = \sum_{u \in C_i} \sum_{v \in C_j} d_{uv}$.

Our 3-approximation algorithm relies on the following observation. Suppose that $C_1^*, \ldots, C_k^*$ is the optimal clustering. Then, we can select centers $c_1, \ldots, c_k$ in $C_1^*, \ldots, C_k^*$ so that the distance between every point $u \in C_i$ and $v \in C_j$ approximately equals $d_{uc_{i \wedge j}} + d_{vc_{i \wedge j}}$, where $i \wedge j = \min(i, j)$. Our approximation algorithm finds centers $c_1, \ldots, c_k$ by trying all possible combinations and obtains an approximate solution by finding the minimum cost assignment of points in V to centers $c_1, \ldots, c_k$.

First, we prove Lemmas B.1 and B.2 that upper- and lower-bound the cost of a clustering in terms of its *clustering profile*, which is defined as follows. Consider a set of points $c_1, \ldots, c_k \in V$ and natural numbers $n_1, \ldots n_k$ that add up to n. We say that the set of pairs $\mathbf{\Pi} = \{(c_i, n_i)\}_{i=1}^k$ is a clustering profile. Let

$$F_{\mathbf{\Pi}}(u, i) = 2 \sum_{j=1}^{k} n_j \, A_{ij} \, d_{uc_{i \wedge j}}. \tag{4}$$

Lemma B.1. *For every partitioning $C_1, \ldots, C_k$ of V, and an arbitrary set of centers $c_1, \ldots, c_k \in V$, we have*

$$\textsc{cost}(C_1, \ldots, C_k) \le \sum_{i=1}^{k} \sum_{u \in C_i} F_{\mathbf{\Pi}}(u, i), \tag{5}$$

where $F_{\mathbf{\Pi}}(u, i)$ is defined as above, $n_i = |C_i|$, and $\mathbf{\Pi}$ is the set of pairs (c_i, n_i).

Proof. For $u \in C_i$ and $v \in C_j$, we have $d_{uv} \le d_{uc_{i \wedge j}} + d_{c_{i \wedge j} v}$. Thus,

$$\mathrm{W}(C_i, C_j) = \sum_{u \in C_i} \sum_{v \in C_j} d_{uv} \le \sum_{u \in C_i} \sum_{v \in C_j} \left(d_{uc_{i \wedge j}} + d_{c_{i \wedge j} v}\right) = 2 \sum_{u \in C_i} \sum_{v \in C_j} d_{uc_{i \wedge j}} = 2 n_j \sum_{u \in C_i} d_{uc_{i \wedge j}}.$$

Substituting this into cost, we get

$$\textsc{cost}(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{j=1}^{k} A_{ij} \mathrm{W}(C_i, C_j) \le \sum_{i=1}^{k} \sum_{j=1}^{k} A_{ij} 2 n_j \sum_{u \in C_i} d_{uc_{i \wedge j}} = \sum_{i=1}^{k} \sum_{u \in C_i} F_{\mathbf{\Pi}}(u, i). \qquad \square$$

Lemma B.2. *For every partitioning $C_1, \ldots, C_k$ of V, there exist centers $c_1 \in C_1, \ldots, c_k \in C_k$ such that*

$$\sum_{i=1}^{k} \sum_{u \in C_i} F_{\mathbf{\Pi}}(u, i) \le 3 \cdot \textsc{cost}(C_1, \ldots, C_k), \tag{6}$$

where $F_{\mathbf{\Pi}}(u, i)$ is defined as above, $n_i = |C_i|$, and $\mathbf{\Pi}$ is the set of pairs (c_i, n_i).

Proof. Consider a random set of centers $c_1, \ldots, c_k$, where each c_i is selected uniformly and independently in C_i. Let $\mathbf{\Pi} = \{(c_i, n_i)\}$. We show that

$$\mathbb{E}_{\mathbf{\Pi}} \left[\sum_{i=1}^{k} \sum_{u \in C_i} F_{\mathbf{\Pi}}(u, i) \right] \le 3 \cdot \textsc{cost}(C_1, \ldots, C_k).$$

Consequently, for some realization of $c_1, \dots, c_k$, inequality (6) holds. Let us represent $F_{\mathbf{\Pi}}(u,i)$ in the following form:

$$F_{\mathbf{\Pi}}(u,i) = \underbrace{\left[n_i\, A_{ii}\, d_{uc_i} + 2 \sum_{j=1}^{i-1} n_j\, A_{ij}\, d_{uc_j} \right]}_{F^{\mathrm{I}}_{\mathbf{\Pi}}(u,i)} + \underbrace{\left[n_i\, A_{ii}\, d_{uc_i} + 2 \sum_{j=i+1}^{k} n_j\, A_{ij}\, d_{uc_i} \right]}_{F^{\mathrm{II}}_{\mathbf{\Pi}}(u,i)}.$$

We separately upper-bound $\mathbb{E}_{\mathbf{\Pi}}\left[\sum_{i=1}^{k} \sum_{u \in C_i} F^{\mathrm{I}}_{\mathbf{\Pi}}(u,i)\right]$ and $\mathbb{E}_{\mathbf{\Pi}}\left[\sum_{i=1}^{k} \sum_{u \in C_i} F^{\mathrm{II}}_{\mathbf{\Pi}}(u,i)\right]$.

I. By definition of COST,

$$\begin{aligned}
\textsc{cost}(C_1, \dots, C_k) &= \sum_{i=1}^{k} \sum_{j=1}^{k} \sum_{u \in C_i} \sum_{v \in C_j} A_{ij}\, d_{uv} \\
&= \sum_{i=1}^{k} \sum_{u \in C_i} \left[\sum_{v \in C_i} A_{ii}\, d_{uv} + 2 \sum_{j=1}^{i-1} \sum_{v \in C_j} A_{ij}\, d_{uv} \right].
\end{aligned}$$

Let us fix $i \in [k]$ and $u \in C_i$. Since c_j is a random point in C_j, we have $\sum_{v \in C_j} A_{ij}\, d_{uv} = n_j \mathbb{E}_{\mathbf{\Pi}}\left[A_{ij}\, d_{uc_j}\right]$. Hence,

$$\sum_{v \in C_i} A_{ii}\, d_{uv} + 2 \sum_{j=1}^{i-1} \sum_{v \in C_j} A_{ij}\, d_{uv} = n_i \mathbb{E}_{\mathbf{\Pi}}\left[A_{ii}\, d_{uc_i}\right] + 2 \sum_{j=1}^{i-1} n_j \mathbb{E}_{\mathbf{\Pi}}\left[A_{ij}\, d_{uc_j}\right] = \mathbb{E}_{\mathbf{\Pi}}\left[F^{\mathrm{I}}_{\mathbf{\Pi}}(u,i)\right].$$

Taking the summation over all $i \in [k]$ and $u \in C_i$, we get

$$\mathbb{E}_{\mathbf{\Pi}}\left[\sum_{i=1}^{k} \sum_{u \in C_i} F^{\mathrm{I}}_{\mathbf{\Pi}}(u,i) \right] = \textsc{cost}(C_1, \dots, C_k).$$

II. We have

$$F^{\mathrm{II}}_{\mathbf{\Pi}}(u,i) = n_i\, A_{ii}\, d_{uc_i} + 2 \sum_{j=i+1}^{k} n_j\, A_{ij}\, d_{uc_i}.$$

For every $j > i$ and $v \in C_j$, we upper bound d_{uc_i} using the triangle inequality $d_{uc_i} \le d_{uv} + d_{vc_i}$. We have

$$\begin{aligned}
\mathbb{E}_{\mathbf{\Pi}}\left[\sum_{u \in C_i} n_j\, A_{ij}\, d_{uc_i} \right] &\le \mathbb{E}_{\mathbf{\Pi}}\left[\sum_{u \in C_i} \sum_{v \in C_j} A_{ij}\, (d_{uv} + d_{vc_i}) \right] \\
&= \frac{1}{n_i} \sum_{u' \in C_i} \sum_{u \in C_i} \sum_{v \in C_j} A_{ij} \big(d_{uv} + d_{vu'} \big) \\
&= 2 \sum_{u \in C_i} \sum_{v \in C_j} A_{ij}\, d_{uv}.
\end{aligned}$$

Hence,

$$\mathbb{E}_{\mathbf{\Pi}}\left[\sum_{i=1}^{k} \sum_{u \in C_i} F^{II}_{\mathbf{\Pi}}(u,i) \right] \le 2 \sum_{i=1}^{k} \sum_{u \in C_i} \left[\frac{1}{2} \sum_{v \in C_i} A_{ii}\, d_{uv} + \sum_{j > i} \sum_{v \in C_j} A_{ij}\, d_{uv} \right] = 2 \cdot \textsc{cost}(C_1, \dots, C_k).$$

Combining bounds for $F^{\mathrm{I}}_{\mathbf{\Pi}}$ and $F^{\mathrm{II}}_{\mathbf{\Pi}}$, we get

$$\mathbb{E}_{\mathbf{\Pi}}\left[\sum_{i=1}^{k}\sum_{u\in C_i} F_{\mathbf{\Pi}}(u,i)\right] \le \textsc{cost}(C_1,\dots,C_k) + 2\cdot\textsc{cost}(C_1,\dots,C_k) = 3\cdot\textsc{cost}(C_1,\dots,C_k). \quad \square$$

We now use Lemma B.1 and Lemma B.2 to get a 3-approximation for Weighted Metric Clustering.

Theorem B.3. *For every* fixed *integer $k > 1$, there exists a polynomial-time 3-approximation algorithm for Weighted Metric Clustering.*

Proof. Let $C_1^*,\dots,C_k^*$ be the optimal solution for the problem. Our algorithm guesses the sizes of the optimal clusters $\{n_i\}$ and their centers $\{c_i\}$ such that

$$\sum_{i=1}^{k}\sum_{u\in C_i} F_{\mathbf{\Pi}}(u,i) \le 3\cdot\textsc{cost}(C_1^*,\dots,C_k^*). \tag{7}$$

The existence of such centers $c_1,\dots,c_k$ is guaranteed by Lemma 6. The algorithm then finds the minimum cost matching between points in V and k clusters $C_1,\dots,C_k$. Every cluster C_i is matched with n_i points. The cost of assigning point u to cluster C_i equals $F_{\mathbf{\Pi}}(u,i)$. The obtained set of clusters $C_1,\dots,C_k$ is a solution for the Minimum Kernel Clustering Problem.

Observe that if we assigned every point u to the cluster it belongs to in the optimal solution, then the cost of the matching would be upper bounded by $3\cdot\textsc{cost}(C_1^*,\dots,C_k^*)$ (see Equation (7)). Thus, the cost of the optimal matching is also upper bounded by $3\cdot\textsc{cost}(C_1^*,\dots,C_k^*)$. By Lemma 5, we have

$$\textsc{cost}(C_1,\dots,C_k) \le \sum_{i=1}^{k}\sum_{u\in C_i} F_{\mathbf{\Pi}}(u,i) \le 3\cdot\textsc{cost}(C_1^*,\dots,C_k^*). \quad \square$$

C Proof of Theorem 4.2

We now describe an algorithm for transforming local distributions into nearly independent local distributions. Our algorithm is based on Algorithm 4.4 from the paper by Raghavendra and Tan [2012]. Note that the running time of the algorithm is exponential in δ, γ, η and k.

Algorithm 2 provides a pseudocode for our algorithm, which iteratively updates distribution $\mathbb{P}$ until it is (γ,δ)-independent. Initially, $\mathbb{P}^{(0)} = \mathbb{P}$. At every step $t \in \{0,\dots,r-3\}$, the algorithm checks if $\mathbb{P}^{(t)}$ is already (γ,δ)-independent. If it is, the algorithm returns $\{\mathbb{P}^{(t)}_{uv}\}$, and we say that the algorithm *succeeded*.

On the other hand, if $\mathbb{P}^{(t)}$ is not (γ,δ)-independent, we consider two cases. If $r < t-2$, the algorithm finds point w_t and set D_{s_t} that violate the (γ,δ)-independence requirement, i.e.

$$|\{v \in D_{s_t} : \|\mathbb{P}_{w_t}\otimes\mathbb{P}^*_v - \mathbb{P}^*_{w_t,v}\|_{TV} > \delta\}| > \gamma|D_{s_t}|$$

It then assigns w_t to a cluster based on $\mathbb{P}_{w_t}$. Formally, it then picks a random label $\ell_t \in [k]$ with probability $\mathbb{P}_{w_t}[w_t \in C_{j_t}]$ and defines new local distributions $\mathbb{P}^{(t+1)}$ as follows: for $\mathbf{v} \in V^{r-t-1}$ and $\mathbf{b} \in [k]^{r-t-1}$,

$$\mathbb{P}^{(t+1)}_{\mathbf{v}}\left[\mathbf{v}_1 \in C_{\mathbf{b}_1},\dots,\mathbf{v}_{r-t-1} \in C_{\mathbf{b}_{r-t-1}}\right] = \frac{\mathbb{P}^{(t)}_{\mathbf{v}}\left[\mathbf{v}_1 \in C_{\mathbf{b}_1},\dots,\mathbf{v}_{r-t-1} \in C_{\mathbf{b}_{r-t-1}}, w_t \in C_{\ell_t}\right]}{\mathbb{P}^{(t)}_{\mathbf{v}}[w_t \in C_{\ell_t}]},$$

The expression on the right-hand side is the conditional probability of the event $\mathbf{v}_1 \in C_{\mathbf{b}_1}, \ldots, \mathbf{v}_{r-t-1} \in C_{\mathbf{b}_{r-t-1}}$ given that $w_t \in C_{j_t}$. We denote this conditional probability by $\mathbb{P}^{(t)}_{\mathbf{v}}\left[\mathbf{v}_1 \in C_{\mathbf{b}_1}, \ldots, \mathbf{v}_{r-t-1} \in C_{\mathbf{b}_{r-t-1}} \mid w_t \in C_{\ell_t}\right]$.

If the algorithm didn't reach an (γ, δ)-independent distribution after $r-2$ iterations, the algorithm returns $\{\mathbb{P}^{(r-2)}_{uv}\}$, and we say that the algorithm *failed*.

We restate Theorem 4.2, which analyzes our algorithm.

Theorem 4.2. *For every $\delta, \gamma, \eta \in (0,1)$ and integer $k > 1$, there exists a randomized polynomial-time procedure that given a solution $\mathbb{P}$ to the Sherali–Adams relaxation with $r \geq 2 + \frac{k \log_2 k}{2\delta^2 \gamma \eta}$ rounds, outputs a family of local probability distributions $\{\mathbb{P}^*_u\}_u$ and $\{\mathbb{P}^*_{uv}\}_{uv}$ and exit status ("success" or "failure") such that*

1. *If the algorithm succeeds, then $\mathbb{P}^*$ is (γ, δ)-nearly independent.*
2. *For all $u, v \in V$ and $i, j \in \{1, \ldots, k\}$,*
$$\mathbb{E}\left[\mathbb{P}^*_u[u \in C_i]\right] = \mathbb{P}_u[u \in C_i]$$
$$\mathbb{E}\left[\mathbb{P}^*_{uv}[u \in C_i, v \in C_j]\right] = \mathbb{P}_{uv}[u \in C_i, v \in C_j].$$
3. *The algorithm fails with probability at most η.*

Proof. We now analyze Algorithm 2. If the algorithm succeeds, then the resulting family of local distributions $\mathbb{P}^* = \mathbb{P}^{(t)}$ is (γ, δ)-independent. Thus, we need to show items (2) and (3). Item (2) holds because $\mathbb{P}^{(t)}_u$ and $\mathbb{P}^{(t)}_{uv}$ are (as we prove below) martingales. In fact, one can think of $\mathbb{P}^{(t)}_u$ and $\mathbb{P}^{(t)}_{uv}$ as of Doob martingales. Indeed,

$$\mathbb{E}\left[\mathbb{P}^{(t+1)}_u[u \in C_i] \mid \mathbb{P}^{(t)}\right] = \sum_{j=1}^{k} \frac{\mathbb{P}^{(t)}_{u,w_t}[u \in C_i;\, w_t \in C_j]}{\mathbb{P}_{u,w_t}[w_t \in C_j]} \Pr[\ell_t = j \mid \mathbb{P}^{(t)}].$$

Using that $\Pr[\ell_t = j \mid \mathbb{P}^{(t)}] = \mathbb{P}_{w_t}[w_t \in C_j] = \mathbb{P}_{u,w_t}[w_t \in C_j]$, we get

$$\mathbb{E}\left[\mathbb{P}^{(t+1)}_u[u \in C_i] \mid \mathbb{P}^{(t)}\right] = \sum_{j=1}^{k} \mathbb{P}^{(t)}_{u,w_t}[u \in C_i;\, w_t \in C_j] = \mathbb{P}^{(t)}_u[u \in C_i].$$

Similarly,

$$\mathbb{E}\left[\mathbb{P}^{(t+1)}_{uv}[u \in C_i; v \in C_j] \mid \mathbb{P}^{(t)}\right] = \mathbb{P}^{(t)}_{uv}[u, v \in C_i].$$

We next bound the probability that algorithm MakeIndependent fails. To this end, we define the following function:

$$\Phi(t) = \frac{1}{k} \sum_{s=1}^{k} \mathrm{Avg}_{v \in D_s} \left(- \sum_{i=1}^{k} \mathbb{P}^{(t)}_v[v \in C_i] \log_2 \mathbb{P}^{(t)}_v[v \in C_i] \right).$$

Observe that the expression in the round brackets above is the entropy of the distribution $\mathbb{P}_v$. It is always non-negative. It is also upper bounded by $\log_2 k$ because each v can take k distinct values – labels for sets $C_1, \ldots, C_k$. Thus, $\Phi(t) \leq \log_2 k$. Next, we will show that at every step of MakeIndependent, $\Phi(t)$ is decreased by at least $2\delta^2\gamma/k$ in expectation. Let

$$\Delta\Phi(t) = \Phi(t) - \Phi(t+1).$$

Lemma C.1. *If the algorithm does not succeed by step t (where $t < r-2$), then*

$$\mathbb{E}\left[\Delta\Phi(t) \mid \mathbb{P}^{(t)}\right] \geq \frac{2\delta^2\gamma}{k}.$$

We first prove an auxiliary claim.

Claim C.2. *Consider two distributed random variables X and Y. Denote their joint distribution by $\mathbb{Q}_{xy}$ and their marginal distributions by $\mathbb{Q}_x$ and $\mathbb{Q}_y$, respectively. Let Y' be an independent random variable having distribution $\mathbb{Q}_y$. Then,*

$$H(X) - \mathbb{E}_{Y'}\left[H(X \mid Y = Y')\right] = D_{KL}\left(\mathbb{Q}_{xy} \parallel \mathbb{Q}_x \otimes \mathbb{Q}_y\right),$$

where $H(X)$ is the entropy of X, $H(X \mid Y = Y')$ is the conditional entropy, and $D_{KL}(\cdot \parallel \cdot)$ is the Kullback—Leibler (KL) divergence.

Proof. Consider the mutual information $I(X;Y)$ of random variables X and Y. On the one hand, $I(X;Y) = H(X) - H(X \mid Y)$, and on the other hand, $I(X,Y) = D_{KL}\left(\mathbb{Q}_{xy} \parallel \mathbb{Q}_x \otimes \mathbb{Q}_y\right)$. Thus,

$$\mathbb{E}_{Y'}\left[H(X \mid Y = Y')\right] = H(X \mid Y) = H(X) - D_{KL}\left(\mathbb{Q}_{xy} \parallel \mathbb{Q}_x \otimes \mathbb{Q}_y\right). \qquad \square$$

Proof of Claim C.1. Consider the t-th step of the algorithm. Suppose that $P^{(t)}$ is not yet (γ,δ)-nearly independent. In this case, the algorithm picks point w_t and set D_{s_t} for which

$$\left|\left\{v \in D_{s_t} : \|\mathbb{P}^{(t)}_{w_t} \otimes \mathbb{P}^{(t)}_{v} - \mathbb{P}^{(t)}_{w_t,v}\|_{TV} > \delta\right\}\right| > \gamma|D_{s_t}|.$$

By Pinsker's inequality, for every v, we have

$$D_{KL}\left(\mathbb{P}^{(t)}_{w_t,v} \parallel \mathbb{P}^{(t)}_{w_t} \otimes \mathbb{P}^{(t)}_{v}\right) \geq 2\|\mathbb{P}^{(t)}_{w_t} \otimes \mathbb{P}^{(t)}_{v} - \mathbb{P}^{(t)}_{w_t,v}\|^2_{TV},$$

Thus, for the chosen w_t and D_{s_t}, we have

$$\left|\left\{v \in D_{s_t} : D_{KL}\left(\mathbb{P}^{(t)}_{w_t,v} \parallel \mathbb{P}^{(t)}_{w_t} \otimes \mathbb{P}^{(t)}_{v}\right) \geq 2\delta^2\right\}\right| > \gamma|D_{s_t}|.$$

Hence,

$$\mathrm{Avg}_{v \in D_{s_t}} D_{KL}\left(\mathbb{P}^{(t)}_{w_t,v} \parallel \mathbb{P}^{(t)}_{w_t} \otimes \mathbb{P}^{(t)}_{v}\right) > 2\delta^2\gamma. \tag{8}$$

We are now ready to bound $\Phi(t)$. Write,

$$\Delta\Phi(t) = \frac{1}{k}\sum_{s=1}^{k} \mathrm{Avg}_{v \in D_s}\left(-\sum_{i=1}^{k} \mathbb{P}^{(t)}_{v}[v \in C_i]\ \log_2 \mathbb{P}^{(t)}_{v}[v \in C_i] + \sum_{i=1}^{k} \mathbb{P}^{(t+1)}_{v}[v \in C_i]\ \log_2 \mathbb{P}^{(t+1)}_{v}[v \in C_i]\right).$$

The expression in the brackets is the difference between the entropy of distribution $\mathbb{P}^{(t)}_{v}$ and $\mathbb{P}^{(t+1)}_{v}$. Distribution $\mathbb{P}^{(t+1)}_{v}$ is the conditional distribution $\mathbb{P}^{(t+1)}_{vw_t}$ given $w_t \in C_i$. Thus, by Claim C.2,

$$-\sum_{i=1}^{k} \mathbb{P}^{(t)}_{v}[v \in C_i]\ \log_2 \mathbb{P}^{(t)}_{v}[v \in C_i] + \sum_{i=1}^{k} \mathbb{P}^{(t+1)}_{v}[v \in C_i]\ \log_2 \mathbb{P}^{(t+1)}_{v}[v \in C_i] \geq D_{KL}\left(\mathbb{P}^{(t)}_{w_t,v} \parallel \mathbb{P}^{(t)}_{w_t} \otimes \mathbb{P}^{(t)}_{v}\right).$$

Consequently,

$$\Delta\Phi(t) \geq \frac{1}{k}\sum_{s=1}^{k} \mathrm{Avg}_{v \in D_s} D_{KL}\left(\mathbb{P}^{(t)}_{w_t,v} \parallel \mathbb{P}^{(t)}_{w_t} \otimes \mathbb{P}^{(t)}_{v}\right).$$

Using inequality (8) and that D_{KL} is always non-negative, we conclude that $\Phi(t) \geq 2\delta^2\gamma/k$. $\square$

Let T be the last step of algorithm MakeIndependent. By Lemma C.1, $\Phi(t) + t \cdot (2\delta^2\gamma/k)$ is a supermartingale. Thus,

$$\mathbb{E}\left[T \cdot \frac{2\delta^2\gamma}{k}\right] \le \mathbb{E}\left[\sum_{i=0}^{T-1} \Delta\Phi(t)\right] = \mathbb{E}\left[\Phi(0) - \Phi(T)\right] \le \log_2 k,$$

here, we are using that T is a stopping time, $\Phi(0) \le \log_2 k$, and $\Phi(T) \ge 0$. We have

$$\mathbb{E}\left[T\right] \le \frac{k \log_2 k}{2\delta^2\gamma}.$$

By Markov's inequality,

$$\Pr\left[T \ge \frac{k \log_2 k}{2\delta^2\gamma\eta}\right] \le \eta. \qquad \square$$

D Main Algorithm

In this section, we present our main algorithm for Metric Kernel Clustering. We provide pseudocode for the algorithm in Figure 1. The algorithm first guesses centers c_i and cluster sizes n_i as described in the previous section. Then, it solves the Sherali–Adams relaxation and obtains local probability distributions $\{\mathbb{P}\}$. It uses these local probability distributions to tentatively assign points in V to clusters $C_1, \dots, C_k$. It also marks some points as outliers and adds them to set O. Finally, the algorithm leaves tentative cluster assignments intact for non-outlier points and assigns new clusters to outlier points.

Algorithm 1 describes the algorithm for finding a partial clustering $X_1, \dots, X_k$ and set of *outliers* O. The algorithm first finds a solution to the Sherali–Adams (SA) relaxation defined in Section 4. Denote the collection of local distributions by $\{\mathbb{P}\}$. Using these distributions, the algorithm identifies a set of candidate points for every cluster C_i:

$$D_i = \{u \in V : \mathbb{P}_u[u \in C_i] \ge \eta\},$$

where η is the parameter of the algorithm. Loosely speaking, D_i is the set of points that are somewhat likely to be assigned to cluster C_i by the Sherali–Adams relaxation. The algorithm then runs function MakeIndependent with parameters $D_1, \dots, D_k$ (see Section 4.1). This function returns a family of new local distributions $\mathbb{P}^*_u$ and $\mathbb{P}^*_{uv}$ that are (γ, δ)-nearly independent (see Definition 4.1) if MakeIndependent does not fail. Our algorithm now independently assigns every point u label $l_u \in \{1, \dots, k\}$ with probability $\mathbb{P}^*_u[u \in C_{l_u}]$. We call this assignment a tentative assignment, and we declare point u an outlier if u *is tentatively assigned to cluster* C_i *(i.e.,* $l_u = i$*) but* $u \notin D_i$. Also, in the unlikely event that MakeIndependent failed, the algorithm marks all points in V as outliers. We denote the set of outliers by O. Now, we make tentative assignments permanent for non-outlier points. Specifically, we let $X_i = \{u \in V \setminus O : l_u = i\}$. We return sets $X_1, \dots, X_k$ and set O.

Analysis. We now analyze the algorithm. We first upper bound the size of X_i. Note that $X_i \subseteq D_i$ (because every point u outside of D_i which is tentatively assigned to cluster i is marked as an outlier; thus, it does not belong to X_i). Claim D.1 implies that $|X_i| \le n_i/\eta$.

Claim D.1. *For every* i*, we have* $|D_i| \le n_i/\eta$.

Table 1: Probabilities and pseudo-probabilities

Notation	Explanation	Randomness
$\{\mathbb{P}\}$	Pseudo-probabilities returned by SA relaxation	Deterministic
$\{\mathbb{P}^*\}$	Pseudo-probabilities after calling MakeIndependent	Random due to rounding in MakeIndependent
Pr	"True" probability of the algorithm	Random due to independent rounding and rounding in MakeIndependent
$\Pr[\cdot \mid \mathbb{P}^*]$	"True" probability conditioned on result of MakeIndependent	For fixed $\mathbb{P}^*$, randomness is due to independent rounding

Proof. For every $u \in D_i$, $\mathbb{P}_u[u \in C_i] \geq \eta$. Thus,

$$|D_i| \leq \sum_{u \in D_i} \frac{\mathbb{P}[u \in C_i]}{\eta} = \frac{n_i}{\eta},$$

where in the last equality we used the linear programming constraint $\sum_{u \in V} \mathbb{P}[u \in C_i] = n_i$. □

Lemma D.2. *For every positive integer k, and $\eta, \gamma, \delta \in (0, 1/k)$, there exists a randomized polynomial-time algorithm that given the optimal solution to the Sherali–Adams relaxation described in Section 4 returns disjoint clusters $X_1, \ldots, X_k$ and set of outliers O (also, disjoint from $X_1, \ldots, X_k$) such that*

1. *for every $u \in V$: $Pr[u \in X_i] \leq \mathbb{P}[u \in C_i]$,*
2. *for every $u \in V$: $\Pr[u \in O] \leq \eta k$,*
3. *the expected cost of non-outliers is at most*

$$\frac{1}{2}\mathbb{E}\Big[\sum_{i,j}\sum_{\substack{u \in X_i \\ v \in X_j}} A_{ij} d_{uv}\Big] \leq OPT_{LP_I} + \frac{3(\gamma+\delta)}{\eta^2} OPT_{LP_{II}} \leq \Big(1 + \frac{3(\gamma+\delta)}{\eta^2}\Big)\, OPT_{SA}. \tag{9}$$

where OPT_{SA} is the cost of the optimal solution to the Sherali–Adams relaxation;

Remark: The running time of the algorithm is polynomial in n and k for every fixed η, γ, δ. In order to obtain a $(1+\varepsilon)$-approximation for the Minimum Kernal Clustering Problem, we will use this lemma with $\eta \approx \varepsilon^2/k^2$ and $\gamma = \delta \approx \varepsilon^5/k^4$.

Proof. **Item 1.** We first prove that $\Pr[u \in X_i] \leq \mathbb{P}[u \in C_i]$ and $\Pr[u \in O] \leq \eta k$. Point u belongs to X_i if $l_u = i$ but $u \notin O$. Thus, $\Pr[u \in X_i] \leq \Pr[l_u = i]$. Since the algorithm assigns label i to l_u with probability $\mathbb{P}^*[u \in C_i]$, we have

$$\Pr[l_u = i] = \mathbb{E}_{\mathbb{P}^*}\big[\Pr[l_u = i \mid \mathbb{P}^*]\big] = \mathbb{E}_{\mathbb{P}^*}\big[\mathbb{P}^*_u[u \in C_i]\big] = \mathbb{P}_u[u \in C_i]$$

The last equality follows from Theorem 4.2, item 2.

Item 2. Point u is an outlier if $l_u = i$, but $u \notin D_i$ (that is, $\mathbb{P}[u \in C_i] < \eta$) or MakeIndependent fails. Thus,

$$\Pr[u \in O] \leq \eta + \sum_{i=1}^{k} \Pr\big[l_u = i\big] \cdot \mathbb{1}\Big\{\mathbb{P}[u \in C_i] < \eta\Big\} = \eta + \sum_{i=1}^{k} \mathbb{P}[u \in C_i] \cdot \mathbb{1}\Big\{\mathbb{P}[u \in C_i] < \eta\Big\}.$$

Observe that each term $\mathbb{P}[u \in C_i] \cdot \mathbb{1}\Big\{\mathbb{P}[u \in C_i] < \eta\Big\}$ is at most η, since the term equals 0 when $\mathbb{P}[u \in C_i] \geq \eta$. Moreover, since $\eta < 1/k$, at least one of the terms equals 0. Hence, $\Pr[u \in O] \leq \eta + \eta(k-1) = \eta k$.

Item 3. We now proceed to show bound (9). We will use that $X_i \subseteq D_i$ for all i. Write

$$\frac{1}{2}\mathbb{E}\Big[\sum_{i,j}\sum_{\substack{u\in X_i\\ v\in X_j}} A_{ij}\, d_{uv}\Big] = \sum_{i,j}\sum_{\substack{u\in D_i\\ v\in D_j}} A_{ij}\, d_{uv}\, \Pr\big[u \in X_i, v \in X_j\big].$$

If $u \in X_i$ and $v \in X_j$, then $l_u = i$ and $l_v = j$. Thus,

$$\begin{aligned}
\Pr\big[u \in X_i,\, v \in X_j\big] &\leq \Pr[l_u = i,\, l_v = j] \\
&= \mathbb{E}_{\mathbb{P}^*}\Big[\Pr\big[l_u = i,\, l_v = j \mid \mathbb{P}^*\big]\Big] && \text{(by Theorem 4.2)} \\
&= \mathbb{E}_{\mathbb{P}^*}\Big[\mathbb{P}^*\big[u \in C_i\big] \cdot \mathbb{P}^*[v \in C_j]\Big]. && \text{(due to independent rounding)}
\end{aligned}$$

Thus,

$$\frac{1}{2}\mathbb{E}\Big[\sum_{i,j}\sum_{\substack{u\in X_i\\ v\in X_j}} A_{ij}d_{uv}\Big] \leq \frac{1}{2}\sum_{i,j}\sum_{\substack{u\in D_i\\ v\in D_j}} A_{ij}d_{uv} \cdot \mathbb{E}_{\mathbb{P}^*}\Big[\mathbb{P}^*\big[u \in C_i\big] \cdot \mathbb{P}^*[v \in C_j]\Big].$$

Let E_δ be the set of all pairs (u, v) that are not δ-nearly independent with respect to $\mathbb{P}^*$. If $(u, v) \notin E_\delta$, then

$$\mathbb{P}^*_u\big[u \in C_i\big] \cdot \mathbb{P}^*_v[v \in C_j] \leq \mathbb{P}^*_{uv}\big[u \in C_i, v \in C_j\big] + \delta.$$

Consequently, for all u and v, we have

$$\mathbb{P}^*_u\big[u \in C_i\big] \cdot \mathbb{P}^*_v[v \in C_j] \leq \mathbb{P}^*_{uv}\big[u \in C_i, v \in C_j\big] + \mathbb{1}\Big\{(u, v) \in E_\delta\Big\} + \delta.$$

Thus,

$$\frac{1}{2}\mathbb{E}\Big[\sum_{i,j}\sum_{\substack{u\in X_i\\ v\in X_j}} A_{ij}d_{uv}\Big] \leq \frac{1}{2}\sum_{i,j}\sum_{\substack{u\in D_i\\ v\in D_j}} A_{ij}d_{uv}\Big(\mathbb{E}_{\mathbb{P}^*}\big[\mathbb{P}^*_{uv}\big[u \in C_i, v \in C_j\big]\big] + \mathbb{1}\Big\{(u, v) \in E_\delta\Big\} + \delta\Big).$$

We split the right-hand side into two terms:

$$\begin{aligned}
\frac{1}{2}\sum_{i,j}\sum_{\substack{u\in X_i\\ v\in X_j}} A_{ij}d_{uv} \leq \frac{1}{2}\sum_{i,j}\sum_{\substack{u\in D_i\\ v\in D_j}} A_{ij}d_{uv}\mathbb{E}_{\mathbb{P}^*}\left[\mathbb{P}^*_{uv}\left[u \in C_i, v \in C_j\right]\right] \\
+ \frac{1}{2}\sum_{i,j}\sum_{\substack{u\in D_i\\ v\in D_j}} A_{ij}d_{uv}\left(\mathbb{1}\Big\{(u, v) \in E_\delta\Big\} + \delta\right).
\end{aligned} \tag{10}$$

In the first term, $\mathbb{E}_{\mathbb{P}^*}\Big[\mathbb{P}^*_{uv}\big[u \in C_i, v \in C_j\big]\Big] = \mathbb{P}_{uv}\big[u \in C_i, v \in C_j\big]$ by Theorem 4.2, item 2. Thus, the first term equals:

$$\begin{aligned}
\frac{1}{2}\sum_{i,j}\sum_{\substack{u\in D_i\\ v\in D_j}} A_{ij}d_{uv}\mathbb{E}_{\mathbb{P}^*}\left[\mathbb{P}^*_{uv}\big[u \in C_i, v \in C_j\big]\right] &= \frac{1}{2}\sum_{i,j}\sum_{\substack{u\in D_i\\ v\in D_j}} A_{ij}d_{uv}\mathbb{P}_{uv}\big[u \in C_i, v \in C_j\big] \\
&\leq \frac{1}{2}\sum_{i,j}\sum_{\substack{u\in V\\ v\in V}} A_{ij}d_{uv}\mathbb{P}_{uv}\big[u \in C_i, v \in C_j\big] \\
&= OPT_{LP_I}.
\end{aligned}$$

Let us bound the second term on the right-hand side of (10), which we denote by R. Using the triangle inequality, we replace d_{uv} with $d_{uc_{i\wedge j}} + d_{c_{i\wedge j}v}$. We have

$$R \le \sum_{i,j} A_{ij} \sum_{(u,v)\in D_i\times D_j} \frac{d_{uc_{i\wedge j}} + d_{c_{i\wedge j}v}}{2} \cdot \Big(\mathbb{1}\Big\{(u,v)\in E_\delta\Big\} + \delta\Big).$$

The expression on the right-hand side is symmetric with respect to u and v. Thus,

$$\begin{aligned} R &\le \sum_{i,j} A_{ij} \sum_{\substack{u\in D_i\\ v\in D_j}} d_{uc_{i\wedge j}} \Big(\mathbb{1}\Big\{(u,v)\in E_\delta\Big\} + \delta\Big) \\ &= \sum_{i,j} A_{ij} \sum_{u\in D_i} d_{uc_{i\wedge j}} \sum_{v\in D_j} \Big(\mathbb{1}\Big\{(u,v)\in E_\delta\Big\} + \delta\Big). \end{aligned}$$

Observe that $\sum_{v\in D_j} \mathbb{1}\{(u,v)\in E_\delta\} \le \gamma|D_j|$, because $\mathbb{P}^*$ is (γ,δ)-nearly independent distribution and E_δ is the set of pairs (u,v) that are not δ-nearly independent (see Definition 4.1). Thus,

$$R \le \sum_{i,j} A_{ij} \sum_{u\in D_i} d_{uc_{i\wedge j}} (\gamma+\delta)|D_j|.$$

By Claim D.1, $|D_j| \le \frac{n_j}{\eta}$. Therefore,

$$R \le \frac{\gamma+\delta}{\eta} \sum_{i=1}^{k} \sum_{u\in D_i} \Big[\sum_{j=1}^{k} A_{ij} d_{uc_{u\wedge v}} n_j\Big] = \frac{\gamma+\delta}{\eta} \sum_{i=1}^{k} \sum_{u\in D_i} F_\Pi(u,i),$$

where $F_\Pi(u,i)$ is defined in Equation (4).

For every $u\in D_i$, we have $\mathbb{P}[u\in C_i] \ge \eta$ and hence $\frac{1}{\eta}\mathbb{P}[u\in C_i] \ge 1$. Thus,

$$R \le \frac{\gamma+\delta}{\eta^2} \sum_{i=1}^{k} \sum_{u\in D_i} F_\Pi(u,i)\,\mathbb{P}[u\in C_i] \le \frac{\gamma+\delta}{\eta^2} \sum_{i=1}^{k} \sum_{u\in V} F_\Pi(u,i)\,\mathbb{P}[u\in C_i] = \frac{3(\gamma+\delta)}{\eta^2} OPT_{LP_{II}}.$$

This concludes the proof of Lemma D.2. □

E Outlier Assignment Algorithm

In the second phase, the algorithm assigns outliers to sets $Y_1,\dots,Y_k$. It places every outlier point $u\in O$ in set Y_i with probability $\mathbb{P}[u\in C_i]$. Note that the main algorithm (described in the previous section) uses local probability distributions $\mathbb{P}^*_u$ rather than $\mathbb{P}_u$ for assigning points to clusters. The algorithm outputs clustering $X_1\cup Y_1,\dots,X_k\cup Y_k$ if $|X_i| \le \frac{10k\,n_i}{\varepsilon}$ and $|Y_i| \le \frac{10k\,\eta n_i}{\varepsilon}$ for each $i\in\{1,\dots,k\}$. Otherwise, if $|X_i| > \frac{10k\,n_i}{\varepsilon}$ or $|Y_i| > \frac{10k\,\eta n_i}{\varepsilon}$ for some $i\in\{1,\dots,k\}$, the algorithm fails. In this case, we run the 3-approximation algorithm described in Section B and return the obtained clustering.

Claim E.1. *The following bound holds:* $\Pr[u\in Y_i] \le \eta k\,\mathbb{P}[u\in C_i]$.

Proof. By Lemma D.2, $\Pr[u\in O] \le \eta k$. Thus,

$$\Pr[u\in Y_i] = \underbrace{\Pr[u\in Y_i \mid u\in O]}_{=\mathbb{P}_u[u\in C_i]} \cdot \underbrace{\Pr[u\in O]}_{\le\eta} \le \eta k\,\mathbb{P}[u\in C_i]. \tag{11}$$

□

Lemma E.2. *The algorithm fails with probability at most $\varepsilon/5$.*

Proof. By Lemma D.2, $\Pr[u \in X_i] \le \mathbb{P}[u \in C_i]$ and by Claim D.1, $\Pr[u \in Y_i] \le \eta k \mathbb{P}[u \in C_i]$ for every $u \in V$. Thus,

$$\mathbb{E}|X_i| = \sum_{u \in V} \Pr[u \in X_i] \le \sum_{u \in V} \mathbb{P}[u \in C_i] = n_i,$$
$$\mathbb{E}|Y_i| = \eta k \sum_{u \in V} \Pr[u \in Y_i] \le \eta k \sum_{u \in V} \Pr[u \in Y_i] = \eta k\, n_i.$$

By Markov's inequality $\Pr\left[|X_i| > \frac{10k\, n_i}{\varepsilon}\right] < \frac{\varepsilon}{10k}$ and $\Pr\left[|Y_i| > \frac{10k^2\, n_i}{\varepsilon}\right] < \frac{\varepsilon}{10k}$. Thus, by the union bound:

$$\Pr\left[|X_i| > \frac{10k\, n_i}{\varepsilon} \text{ or } |Y_i| > \frac{10k^2\, n_i}{\varepsilon} \text{ for some } i\right] \le \frac{\varepsilon}{5}. \qquad \square$$

Denote the event – "algorithm succeeds" – by $\mathcal{S}$ and the indicator of this event by $\mathbf{1}[\mathcal{S}]$. By Lemma E.2, we have $\Pr[\mathcal{S}] \ge 1 - \varepsilon/5$. We now bound the expected cost of clustering when the algorithm succeeds.

Theorem E.3. *The expected cost of the clustering $X_1 \cup Y_1, \dots, X_k \cup Y_k$ is at most*

$$OPT_{LP_I} + \left(\frac{3(\gamma+\delta)}{\eta^2} + \frac{30\eta k^2}{\varepsilon}\right) OPT_{LP_{II}}.$$

Proof. The expected cost of clustering $X_1 \cup Y_1, \dots X_k \cup Y_k$ equals:

$$\frac{1}{2}\mathbb{E}\left[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u \in X_i \cup Y_i \\ v \in X_j \cup Y_j}} A_{ij} d_{uv}\right] = \frac{1}{2}\mathbb{E}\left[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u \in X_i \\ v \in X_j}} A_{ij} d_{uv}\right] \tag{12}$$
$$+ \mathbb{E}\left[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u \in X_i \\ v \in Y_j}} A_{ij} d_{uv}\right] + \frac{1}{2}\mathbb{E}\left[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u \in Y_i \\ v \in Y_j}} A_{ij} d_{uv}\right].$$

By Lemma D.2, the first term on the right-hand side is bounded by $OPT_{LP_I} + \frac{3(\gamma+\delta)}{\eta^2} OPT_{LP_{II}}$. We now upper bound the second term. To this end, we prove an analog of Lemma B.1.

Claim E.4. *We have*

$$\sum_{i,j} \sum_{\substack{u \in X_i \\ v \in Y_j}} A_{ij} d_{uv} \le \sum_{i,j=1}^{k} A_{ij} \left[\sum_{u \in X_i} |Y_j|\, d_{u c_{i \wedge j}} + \sum_{v \in Y_j} |X_i|\, d_{v c_{i \wedge j}}\right]. \tag{13}$$

Proof. For $u \in C_i$ and $v \in C_j$, we have $d_{uv} \le d_{u c_{i \wedge j}} + d_{c_{i \wedge j} v}$. Thus,

$$\sum_{i,j=1}^{k} \sum_{\substack{u \in X_i \\ v \in Y_j}} A_{ij} d_{uv} \le \sum_{i,j=1}^{k} \sum_{\substack{u \in X_i \\ v \in Y_j}} A_{ij} \left(d_{u c_{i \wedge j}} + d_{c_{i \wedge j} v}\right)$$
$$= \sum_{i,j=1}^{k} \sum_{\substack{u \in X_i \\ v \in Y_j}} A_{ij}\, d_{u c_{i \wedge j}} + \sum_{i,j=1}^{k} \sum_{\substack{u \in X_i \\ v \in Y_j}} A_{ij}\, d_{v c_{i \wedge j}}$$
$$= \sum_{i,j=1}^{k} \sum_{u \in X_i} |Y_j|\, A_{ij}\, d_{u c_{i \wedge j}} + \sum_{i,j=1}^{k} \sum_{v \in Y_j} |X_j|\, A_{ij}\, d_{v c_{i \wedge j}}. \qquad \square$$

If the algorithm succeeds, then for each j, $|X_j| \le \frac{10kn_j}{\varepsilon}$ and $|Y_j| \le \frac{10\eta k^2 n_j}{\varepsilon}$. Thus,

$$\begin{aligned}
\mathbb{E}\Big[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u\in X_i \\ v\in Y_j}} A_{ij}\, d_{uv}\Big] &\le \sum_{i,j=1}^{k} A_{ij}\, \mathbb{E}\Big[\mathbb{1}\{\mathcal{S}\} \sum_{u\in X_i} |Y_j|\, d_{uc_{i\wedge j}} + \mathbb{1}\{\mathcal{S}\} \sum_{u\in Y_j} |X_i| d_{uc_{i\wedge j}}\Big] \\
&\le \sum_{i,j=1}^{k} A_{ij}\, \mathbb{E}\Big[\sum_{u\in X_i} \frac{10\eta k n_j}{\varepsilon}\, d_{uc_{i\wedge j}} + \sum_{u\in Y_i} \frac{10kn_j}{\varepsilon}\, d_{uc_{i\wedge j}}\Big] \\
&= \frac{10k}{\varepsilon} \sum_{i,j=1}^{k} n_j A_{ij} \Big[\sum_{u\in V} (\eta k \Pr[u\in X_i] + \Pr[u\in Y_i])\, d_{uc_{i\wedge j}}\Big].
\end{aligned}$$

Now, by Lemma D.2, $\Pr[u \in X_i] \le \mathbb{P}[u \in C_i]$ and by Claim E.1 $\Pr[u \in Y_i] \le \eta k\, \mathbb{P}[u \in C_i]$. Thus:

$$\begin{aligned}
\mathbb{E}\Big[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u\in X_i \\ v\in Y_j}} A_{ij}\, d_{uv}\Big] &\le \frac{10k}{\varepsilon} \sum_{i,j=1}^{k} n_j A_{ij} \Big[\sum_{u\in V} 2\eta k\, \mathbb{P}_u[u\in C_i]\, d_{uc_{i\wedge j}}\Big] \\
&= \frac{20\eta k^2}{\varepsilon} \sum_{i=1}^{k} \sum_{u\in V} \mathbb{P}_u[u\in C_i] \Big[\sum_{j=1}^{k} A_{ij} n_j\, d_{uc_{i\wedge j}}\Big] \\
&= \frac{20\eta k^2}{\varepsilon} \sum_{i=1}^{k} \sum_{u\in V} \mathbb{P}_u[u\in C_i]\, F_\Pi(u,i) \\
&= \frac{20\eta k^2}{\varepsilon}\, OPT_{LP_{II}},
\end{aligned}$$

where $F_\Pi(u,i)$ is defined in Equation (4).

We now bound the third term in (12). Using the triangle inequality and then rearranging terms as in Claim E.4, we get

$$\frac{1}{2}\mathbb{E}\Big[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u\in Y_i \\ v\in Y_j}} A_{ij} d_{uv}\Big] \le \mathbb{E}\Big[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j=1}^{k} \sum_{u\in Y_i} |Y_j|\, A_{ij}\, d_{uc_{i\wedge j}}\Big].$$

We then replace $|Y_j|$ with an upper bound of $10\eta k n_j/\varepsilon$ and use that $\Pr[u \in Y_i] \le \eta k\, \mathbb{P}[u \in C_i]$ (Claim E.1),

$$\begin{aligned}
\frac{1}{2}\mathbb{E}\Big[\mathbb{1}\{\mathcal{S}\} \cdot \sum_{i,j} \sum_{\substack{u\in Y_i \\ v\in Y_j}} A_{ij} d_{uv}\Big] &\le \frac{10\eta k}{\varepsilon} \mathbb{E}\Big[\sum_{i,j=1}^{k} \sum_{u\in Y_i} n_j\, A_{ij}\, d_{uc_{i\wedge j}}\Big] \\
&= \frac{10\eta k}{\varepsilon} \sum_{i,j=1}^{k} \sum_{u\in V} n_j\, A_{ij}\, d_{uc_{i\wedge j}} \cdot \Pr[u\in Y_i] \\
&\le \frac{10\eta k}{\varepsilon} \sum_{i,j=1}^{k} \sum_{u\in V} n_j\, A_{ij}\, d_{uc_{i\wedge j}} \cdot \eta k\, \mathbb{P}_u[u\in C_i] \\
&= \frac{10\eta^2 k^2}{\varepsilon} \sum_{i=1}^{k} \sum_{u\in V} \mathbb{P}_u[u\in C_i]\, F_\Pi(u,i) \\
&= \frac{10\eta^2 k^2}{\varepsilon} OPT_{LP_{II}}.
\end{aligned}$$

Since $\eta < 1$, the right hand side is less than $\frac{10\eta k^2}{\varepsilon} OPT_{LP_{II}}$. This concludes the proof. □

F Application: Hierarchical Clustering (HC)

We showcase how our general Metric Kernel Clustering framework ($\star$) can be applied to a problem where the goal is to find a hierarchy over clusters rather than a partition. In *Hierarchical Clustering* (HC), given a set of points V, the goal is to bijectively map the points on the leaves of a tree $\mathcal{T}$. Recall from Section 1, our concrete example with matrix A_3 can be thought of as embedding the n points on a ring. Here, we want to instead embed the n points to the leaves of a binary tree; and to do so, we will pick an appropriate matrix A, and show why it allows us to get a PTAS.

Objectives for HC. There are several applications where we want to split data points into hierarchies. Despite the rich literature on algorithmic methods (bottom-up agglomerative or top-down divisive) to produce such hierarchies, there was a general lack of optimization desiderata in HC. Dasgupta [2016] first defined an objective based on graph similarities and proved formal approximation guarantees for HC. This paved the way for a flurry of works in HC [Moseley and Wang, 2017, Charikar and Chatziafratis, 2017, Cohen-Addad, Kanade, Mallmann-Trenn, and Mathieu, 2019, Alon, Azar, and Vainstein, 2020] providing a better understanding to the theoretical underpinnings of HC.

All of the proposed objectives in these works relied on the notion of a lowest common ancestor (LCA) between two nodes. For two leaves i, j, let $\text{LCA}_{\mathcal{T}}(i, j)$ be the LCA of i and j in tree $\mathcal{T}$. The objectives used the *size* of LCA as a penalty factor for the cost of separating an edge; for example, Dasgupta's objective was to minimize the following expression over all binary trees $\mathcal{T}$:

$$cost(\mathcal{T}) = \sum_{u<v} w_{uv} |\text{LCA}_{\mathcal{T}}(u, v)|$$

For the motivation why this may be a good objective we refer the reader to Dasgupta [2016], Cohen-Addad, Kanade, Mallmann-Trenn, and Mathieu [2019]. At a high-level, since the weight w_{ij} denotes similarities, we would like to preserve high similarity edges for as long as possible in the hierarchical tree $\mathcal{T}$, which means that ideally we should cut the edge i, j at the bottom levels of the tree. These levels correspond to small-sized LCA. Notice that the term $|\text{LCA}_{\mathcal{T}}(i, j)|$ is a number between 2 (lowest level of $\mathcal{T}$) and n (at the root of the tree). Minimizing the cost tries to preserve similar endpoints together and split only when their LCA is small.

HC Objective based on Depth. Instead of using the *size* of the LCA, here we study an optimization goal for HC where the penalty term is defined based on the *depth* of the LCA. For a node $i \in \mathcal{T}$, let $h(i)$ denote the height of i in the tree, defined as the number of edges on the shortest path from the root node and i (e.g. $h(i) = 0$ if i is the root node). Our objective is to minimize the following expression over all binary trees $\mathcal{T}$:

$$H(\mathcal{T}) = \sum_{u,v \in V} d_{uv} h(\text{LCA}_{\mathcal{T}}(u, v)) \qquad \text{(Depth-HC)}$$

Our objective corresponds to the fact that in a hierarchical tree representation of $d(\cdot, \cdot)$ we would like to separate the points that are far from each other early in the hierarchical structure, corresponding to small values of h, because otherwise assigning these points to small subtrees would incur a high cost.

Here, d is a metric, and we shall note that HC has been extensively studied for metric spaces, with the goal of finding the best tree metric to fit the metric in the data [Agarwala, Bafna, Farach, Paterson, and Thorup, 1998, Ailon and Charikar, 2005, Cohen-Addad, Das, Kipouridis, Parotsidis, and Thorup, 2022a], or to speed-up linkage methods via approximate nearest neighbors data structures [Moseley, Vassilvtiskii, and Wang, 2021], or to find a tree that is competitive to k-center objectives for multiple values of k simultaneously [Dasgupta and Long, 2005]. See Appendix F.1.

Generalized Depth-Based HC Objective In objective (Depth-HC), d_{uv} is multiplied by the depth of $\text{LCA}_{\mathcal{T}}(i,j)$. In general, dependence on the depth might be more complicated, i.e. we consider the following objective:

$$H_f(\mathcal{T}) = \sum_{u,v \in V} d_{uv}\, f(h(\text{LCA}_{\mathcal{T}}(i,j))) \qquad \text{(Gen-Depth-HC)}$$

for some fixed function f. A natural question is: for which functions f there exists a PTAS for this objective? We show that, as long as f is monotone and satisfies the condition $\lim_{t\to\infty} \frac{f(t+1)}{f(t)} = 1$, a PTAS exists (see Appendix F.2). Note that many natural functions, including all polynomial functions, satisfy this condition.

Motivation from Graph Compression. When dealing with huge graphs, like the Internet graph or social networks, finding a compact way of representing them is challenging. In fact, even how to *label* the nodes in the graph is not obvious. Different labelings result in different storage requirements. The reason is that how much we can compress a graph depends on how we label its vertices. For example, compressing the Internet graph relies on several observations about how webpages are organized made by Raghavan and Garcia-Molina [2003], Boldi and Vigna [2004]. First, pages that are proximal in the lexicographic ordering (on their URLs) tend to have similar sets of neighbors, and second, many links are intra-domain, and therefore likely to point to pages nearby in the lexicographic ordering. These are the *similarity* and *locality* principles respectively. Analogous observations hold in social networks [Chierichetti, Kumar, Lattanzi, Mitzenmacher, Panconesi, and Raghavan, 2009], as proximal users have similar sets of neighbors, and they have empirically shown that taking such observations into account during vertex labeling and clustering can result in significant memory savings.

An important idea in compression is the *offset trick* for vertex-edge representation: think of a billion-sized graph and an edge connecting vertices a, b. Instead of storing the labels a, b explicitly for this edge, we can store one of the labels, say a, and then the *difference* $b - a$. This simple trick can save large amounts of space; for other examples and benefits of offset representations, see Boldi and Vigna [2004]. The problem of finding the best labelings is often called graph reordering and is a powerful technique to increase the locality of the representations of graphs. Many natural combinatorial optimization problems arise in this context, that mainly have to do with how to order vertices on a line so as to minimize the density of edges across the labeled vertices. For example, the classical NP-hard problem of Minimum Linear Arrangement [Charikar, Hajiaghayi, Karloff, and Rao, 2010] is relevant in these applications. However, since the goal is to minimize storage and storage is measured with the number of bits used, Minimum Logarithmic Arrangement and Minimum Logarithmic Gap Arrangement were proposed in Dhulipala, Kabiljo, Karrer, Ottaviano, Pupyrev, and Shalita [2016] as more accurate formulations of the compression problem. For example, in Minimum Logarithmic Arrangement we try to minimize $\sum_{uv\in E} \log|\pi(u) - \pi(v)|$, whereas in Minimum Logarithmic Gap Arrangement we minimize the logarithm of consecutive gaps. As noted in their paper, all three arrangement problems are quite different and they may have very different optimal solutions, yielding different tradeoffs for compression.

Let's return to our hierarchical clustering formulation (Depth-HC). Notice that the bits needed for vertex and edge representations are related to the depth of a vertex in the hierarchy. A possible encoding for example could be to store the left-right traversal for the path connecting a vertex to the root; making sure that most similar vertices end up in nearby clusters ensures larger benefits from using the abovementioned offset trick. Hence, our objective can be seen as trying to group similar nodes together by explicitly minimizing the bits needed for their representation.

F.1 PTAS for Depth-Based Hierarchical Clustering

Consider a full binary tree $\mathcal{T}_\circ$ of depth $\log(1/\varepsilon)$. Let $\ell = 1/\varepsilon$ be a set of labels corresponding to each of the leaves of this tree. Let the constraints be defined by a weight matrix with entries $A_{ij} = h(\mathrm{LCA}_{\mathcal{T}_\circ}(i,j))$, i.e. if a point a gets assigned a label i and a point b gets assigned a label j in the CSP then the corresponding cost of this pair is $h(\mathrm{LCA}_{\mathcal{T}_\circ}(i,j))$. We denote this CSP as CSP_{HC}. As we show below, finding a $(1+\varepsilon)$-approximate solution to this CSP suffices in order to get a $(1+\varepsilon)$ approximation for the hierarchical clustering objective above.

We next state our main lemma which bounds the total inter-cluster distance.

Lemma F.1. *Consider an algorithm that builds a full binary tree by solving the corresponding* CSP_{HC} *and then assigning the points to the leaves of a full binary tree* $\mathcal{T}_\circ$ *based on their labels in the solution (splitting each leaf into binary subtrees randomly). If the solution to* CSP_{HC} *is* $(1+\varepsilon)$*-approximate then the solution to the hierarchical clustering objective is* $(1+O(\varepsilon))$*-approximate.*

Let γ_i *denote the weight cut exactly at depth* i*. Then for* $t = \log 1/\varepsilon$*:*

$$\sum_{i=t}^{\infty} \gamma_t \leq 16\varepsilon \mathrm{OPT}$$

Using this lemma, we show our main result.

Theorem F.2. *For any metric* d*, there exists a polynomial-time approximation scheme for minimizing the hierarchical clustering objective* $H(\mathcal{T})$*.*

Proof. By Lemma F.1, in the optimal tree, the cost associated with all internal nodes at depth at least $\log 1/\varepsilon$ is only $O(\varepsilon)\mathrm{OPT}$. Hence, by $(1+\varepsilon)$-approximating the CSP_{HC} which describes the cost of the first $\log 1/\varepsilon$ levels in the optimum tree, we get a $(1+O(\varepsilon))$-approximation overall. □

Proof of Lemma F.1. Consider the optimum tree T^* and fix a depth threshold t. First, note that w.l.o.g. we can assume that T^* is a complete binary tree with some leaves potentially empty.

Proposition F.3. *Let* α *be the total weight inside the subtrees at depth* t *and* β *be the cost contributed by all LCA nodes at depth less than* t*, then:*

$$\beta + \alpha t \leq \mathrm{OPT} \leq \beta + \alpha(t+1)$$

Proof. Consider splitting all subtrees at depth t uniformly at random. Since a $1/2^{i-t+1}$-fraction of the weight has the least common ancestor at depth i, the resulting expected cost is:

$$\sum_{i=t}^{\infty} \frac{i\alpha}{2^{i-t+1}} = (t+1)\alpha$$ □

Recall that γ_i denotes the weight cut exactly at depth i. Let S denote the overall weight, i.e. $S = \sum_{i=0}^{\infty} \gamma_i$. Let $\beta_i = \sum_{j=0}^{i-1} j\gamma_j$ and $\alpha_i = S - \sum_{j=0}^{i-1} \gamma_j$. By Proposition F.3,

$$\sum_{j=0}^{i-1} j\gamma_j + i\left(S - \sum_{j=0}^{i-1} \gamma_j\right) \leq \text{OPT} \leq \sum_{j=0}^{i-1} j\gamma_j + (i+1)\left(S - \sum_{j=0}^{i-1} \gamma_j\right).$$

Rearranging the summations:

$$iS + \sum_{j=0}^{i-1}(j-i)\gamma_j \leq \text{OPT} \leq (i+1)S + \sum_{j=0}^{i-1}(j-i-1)\gamma_j$$

Recall that $\text{OPT} = \sum_{j=0}^{\infty} j\gamma_j$ and hence we have:

$$\begin{aligned}\sum_{j=0}^{\infty} j\gamma_j &\leq (i+1)S + \sum_{j=0}^{i-1}(j-i-1)\gamma_j \\ &= (i+1)\sum_{j=0}^{\infty}\gamma_j + \sum_{j=0}^{i-1}(j-i-1)\gamma_j \\ &= \sum_{j=0}^{i-1} j\gamma_j + (i+1)\sum_{j=i}^{\infty}\gamma_j\end{aligned}$$

Rearranging the terms in the above:

$$(i+1)\sum_{j=i}^{\infty}\gamma_j \geq \sum_{j=i}^{\infty} j\gamma_j,$$

or equivalently:

$$\gamma_i \geq \sum_{j=i+1}^{\infty}(j-i-1)\gamma_j$$

In particular, for $i = 1$ this implies that $\gamma_1 \geq \sum_{j=3}^{\infty}(j-2)\gamma_j$.

Proposition F.4. *For every $i \geq 3$ we have $\gamma_1 \geq 2^{i-3}\gamma_i$.*

Proof. As shown above, we have $\gamma_1 \geq \sum_{j=3}^{\infty}(j-2)\gamma_j$, and by induction, we can show an even stronger statement:

$$\gamma_1 \geq 2^{i-3}\sum_{j=i}^{\infty}(j-i+1)\gamma_j$$

The base case $i = 3$ holds by the inequality above. Suppose we have the statement above for i.

Then for $i+1$ just substitute $\gamma_i \geq \sum_{j=i+1}^{\infty}(j-i-1)\gamma_j$ on the RHS, and we have:

$$\begin{aligned}
\gamma_1 &\geq 2^{i-3}\sum_{j=i}^{\infty}(j-i+1)\gamma_j \\
&= 2^{i-3}\sum_{j=i+1}^{\infty}(j-i+1)\gamma_j + 2^{i-3}\gamma_i \\
&\geq 2^{i-3}\sum_{j=i+1}^{\infty}(j-i+1)\gamma_j + 2^{i-3}\sum_{j=i+1}^{\infty}(j-i-1)\gamma_j \\
&= 2^{i-2}\sum_{j=i+1}^{\infty}(j-i)\gamma_j,
\end{aligned}$$

which is the inductive statement for the next index $i+1$. □

Finally, let $t = \log(1/\varepsilon)$. By the proposition we have $\gamma_i \leq 2^{3-i}\gamma_1$. Summing up this inequality from t to ∞ we have:

$$\sum_{i=t}^{\infty}\gamma_i \leq \sum_{i=t}^{\infty}2^{3-i}\gamma_1 = 2^{-t}\gamma_1\sum_{i=t}^{\infty}2^{t-i+3} \leq 16\varepsilon\gamma_1 \leq 16\varepsilon\cdot\mathrm{OPT}.$$ □

F.2 PTAS for the Generalized Depth-Based Objective

Our result from the previous section, which gives a PTAS for $H(\mathcal{T})$, can be generalized to arbitrary functions of depth. In particular, we give a PTAS for the following generalized depth-based objective:

$$H_f(\mathcal{T}) = \sum_{u,v\in V} d_{uv}\, f(h(\mathrm{LCA}_{\mathcal{T}}(u,v))) \qquad \text{(Gen-Depth-HC)}$$

Theorem F.5. *For any monotone non-decreasing function f satisfying $\lim_{t\to\infty}\frac{f(t+1)}{f(t)} = 1$ and a metric space d there exists a polynomial-time approximation scheme for minimizing the hierarchical clustering objective $H_f(\mathcal{T})$.*

Proof. We show that one can truncate the optimal tree at a certain level and then use our PTAS for Weighted Metric Clustering up to this level and then split the resulting clusters randomly at lower levels with only a slight increase in the cost.

Consider the optimal tree $\mathcal{T}^*$. Similarly to the previous section, we can truncate the tree at depth t^*, which gives us 2^{t^*} clusters, and then build a random tree for every resulting cluster. For any u, v from the same cluster, edge (u,v) is cut with probability 2^{-i-1} on level t^*+i, and hence the expected contribution of the edge (x,y) to the objective is

$$d_{uv}\sum_{i=0}^{\infty}\frac{f(t^*+i)}{2^{i+1}}.$$

On the other hand, using that f is monotone, the smallest possible contribution of this edge to the objective is $d_{uv}f(t^*)$. Hence, for any cluster C, the ratio of the expected contribution of the edge in the random tree on C to its contribution in the optimal tree on C is at most $\sum_{i=0}^{\infty}\frac{f(t^*+i)}{2^{i+1}f(t^*)}$. Since

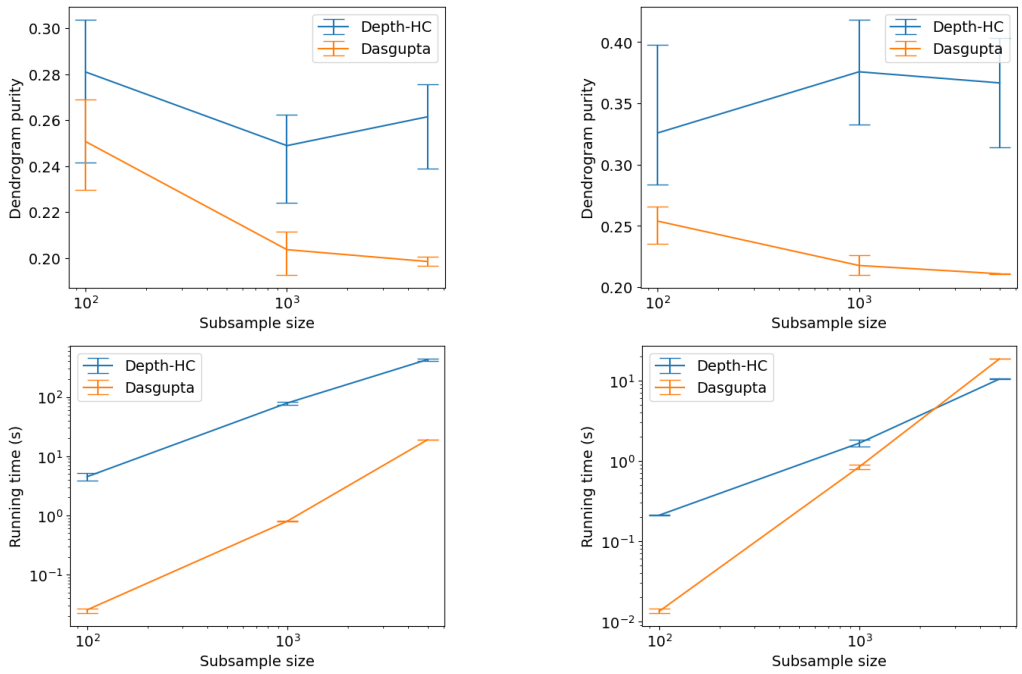

Figure 3: Comparison of the depth-based objective (Depth-HC) and Dasgupta's objective. Left column: ZEBRAFISH GENE COUNTS dataset, right column: AMAZON REVIEWS dataset. Data points correspond to averages over 10 runs, and error bars correspond to the 10% and 90% quantiles. For both datasets, we subsample 100, 1000, and 5000 points.

$\lim_{t\to\infty} \frac{f(t+1)}{f(t)} = 1$, for every ε there exists t_0 such that $f(t+1) < (1+\varepsilon)f(t)$ for all $t \geq t_0$. Hence, for $t^* \geq t_0$:

$$\sum_{i=0}^{\infty} \frac{f(t^*+i)}{2^{i+1} f(t^*)} \leq \sum_{i=0}^{\infty} \frac{(1+\varepsilon)^i}{2^{i+1}} = \frac{1}{2(1-(1+\varepsilon)/2)} = \frac{1}{1-\varepsilon} = 1 + \varepsilon + o(\varepsilon).$$

Hence, there exists t^* – which is a constant depending on f and ε – such that approximating the tree cost up to level t^* suffices for approximating the total tree cost. Finally, similarly to the previous section, we apply our main result on the distance matrix d and the cost matrix $\mathbf{A}$ such that $A_{ij} = f(h_{ij})$, where h_{ij} is the depth of the LCA of clusters with indices i and j. □

G Additional Experiments

We first describe the experimental setup used in Figure 2 from Section 6. The 20 NEWSGROUP dataset used for the experiments consists of the documents. Each document belongs to one of 20 topics, organized in a two-layer hierarchical structure. The first layer of this structure has 7 nodes, which we use to recover the flat clustering of the documents. Finally, to compute distances and similarities between the documents, we convert all the documents to the embedding vectors (after removing redundant information, such as email addresses) using a pre-trained language model initialized as SentenceTransformer('all−MiniLM−L6−v2').

In this section, we present our experiments on additional datasets in the same settings as in Section 6. In Figure 3 experiments, we show the performance of our hierarchical objective (Depth-HC) and the Dasgupta's objective on ZEBRAFISH GENE COUNTS dataset [Wagner et al., 2018] and

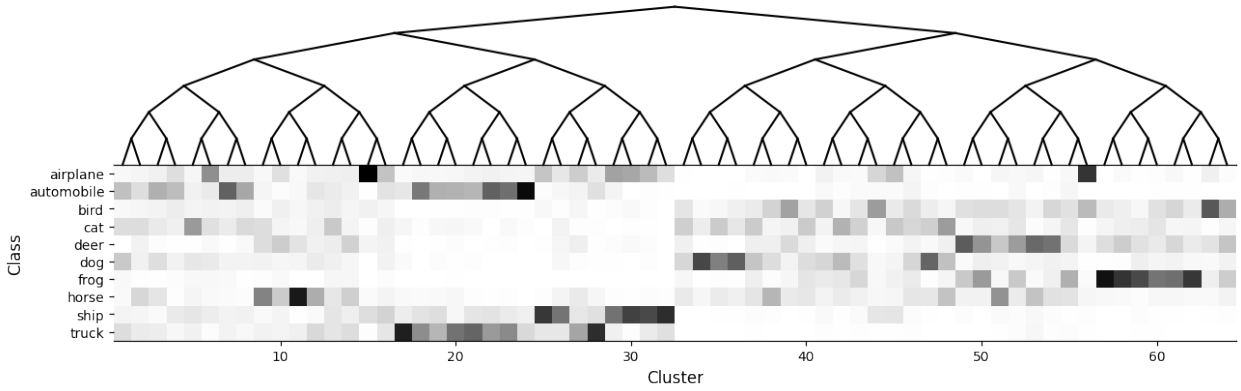

Figure 4: Distribution of classes in the hierarchical clustering for CIFAR-10 dataset. For each class (row), we show its distribution across the clusters (columns) in the hierarchical tree up to depth 6. For each class i and cluster i, define c_{ij} as the number of points in cluster j with class label i. Then, the intensity of (i, j)-th cell's color represents $\frac{c_{ij}}{\sum_{j'} c_{ij'}}$, with black corresponding to value 1, and white corresponding to value 0. Most of the classes are concentrated in small subtrees, which shows that the hierarchical objective achieves good class separation

Amazon reviews dataset[5]. Similarly to Figure 2, (Depth-HC) significantly better recovers the ground-truth clusters, and our algorithm for optimizing (Depth-HC) shows a slower growth rate.

Next, we present our experiments for CIFAR-10 test dataset (10^4 points) and the subset of the Imagenet dataset where from each class we sample a single point (10^3 points).

For CIFAR-10, we build a tree up to depth 6, and in Figure 4 we show the distribution of the classes across the resulting clusters. For most classes, the points from the same class tend to group together in the tree: for example, for class "ship", most of the points lie in clusters 16-23, corresponding to a single subtree. Among these classes, the only ones with substantial intersection are classes "automobiles" and "trucks", which is due to the similarity between these classes.

For the small ImageNet dataset (10^3 images, one image per class), we build a hierarchy up to depth 4 (larger depth provides better separation, but we limit the depth to 4 due to presentation considerations). Then, we select the most common super-classes (for example, "terrier" is a super-class for "Irish terrier", "Tibetan terrier", "silky terrier", etc.) and show the allocation of images corresponding to these super-classes. While the resulting clusters are sometimes mixed, there are clear tendencies: clusters 7 and 12 (counting from left to right) mostly have dogs, cluster 4 has shops and screens, and clusters 9-12 have bears, snakes, and spiders. The mixed results might be attributed to the fact that the similarity between embeddings doesn't precisely represent the similarity between the images (since for classification purposes, it suffices to only separate different classes) or due to the influence of the images outside of the sampled set.

In conclusion, for both datasets, hierarchical clustering according to the new hierarchical clustering objective produces hierarchical trees with good class separation. Finally, the heuristic based on the LP_{II} objective provides shows good empirical performance while being able to handle reasonably-sized datasets.

[5] https://www.kaggle.com/datasets/kashnitsky/hierarchical-text-classification

Figure 5: Sample images from a hierarchy built on the small ImageNet dataset. We build a hierarchy on 10^3 images (one image per class), and we show allocation for the images corresponding to the following super-classes: "terrier", "hound", "snake", "bottle", "shop", "screen", "spider", "bear". For the sake of presentation, we only build a tree up to a depth 4. Empty branches correspond to subtrees without sampled images