Clustering With Center Constraints

Parinya Chalermsook[1] and Suresh Venkatasubramanian[2]

1 **Max-Planck-Institut für Informatik, Germany**
2 **School of Computing, University of Utah, USA**

Abstract

In the classical MAXIMUM INDEPENDENT SET PROBLEM, we are given a graph G of "conflicts" and are asked to find a maximum conflict-free subset. If we think of the remaining nodes as being "assigned" (at unit cost each) to one of these independent vertices and ask for an assignment of minimum cost, this yields the VERTEX COVER problem. In this paper, we consider a more general scenario where the assignment costs might be given by a distance metric d (which can be unrelated to G) on the underlying set of vertices. This problem, in addition to being a natural generalization of vertex cover and an interesting variant of the k-MEDIAN problem, also has connection to constrained clustering and database repair.

Understanding the relation between the conflict structure (the graph) and the distance structure (the metric) for this problem turns out to be the key to isolating its complexity. We show that when the two structures are unrelated, the problem inherits a trivial upper bound from vertex cover and provide an almost matching lower bound on hardness of approximation. We then prove a number of lower and upper bounds that depend on the relationship between the two structures, including polynomial time algorithms for special graphs.

1998 ACM Subject Classification H.3.3 Clustering, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Clustering, vertex cover, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2013.401

1 Introduction

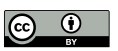
Let $G = (V, E)$ represent a set of *conflicts* between pairs of elements in V and $d : V \times V \to \mathbb{R}$ be a metric capturing the cost of *assigning* one vertex to another. In this paper, we consider the following problem, called *minimum edge-weighted independent set* (MWIS) [18]:

▶ Problem 1.1 (MWIS). Find an independent set of vertices $S \subset V$ such that the *assignment cost* $\sum_{v \in V \setminus S} \min_{s \in S} d(v, s)$ is minimized.

This problem is a natural generalization of VERTEX COVER (when $d(x, y) \equiv 1$ for all $x, y \in V$), and arises naturally in two distinct applications.

Constrained clustering. In a typical application of clustering, the goal is to group close-by objects into a small number of groups. Often the user has domain information about the data in the form of pairs of items that either should be linked together (MUST-LINK) or should not (CANNOT-LINK). Such a clustering problem is called *constrained clustering*[3] and has been studied extensively.

In general, these constraints are provided by users and it can be difficult to make clear judgments about whether two points should be in the same cluster or not, since this depends on the larger context of the clustering. An easier decision is to decide whether two points can serve as cluster *representatives* at the same time or not. Intuitively, if two points are close to each other, then we might expect one or the other to be a cluster center, but not



33rd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013).
Editors: Anil Seth and Nisheeth K. Vishnoi; pp. 401–412

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

both. This is less constraining than a MUST-LINK or CANNOT-LINK constraint: two points connected by a constraint can lie in the same cluster or in different clusters, as long as they are not both cluster centers at the same time.

This center-constrained clustering problem is captured by the above formulation. G captures the constraints, and d is the underlying distance function used for clustering. The goal is now to find a set of cluster centers that are not in conflict such that the total cost of assigning points to cluster centers is minimized (here the cost of a cluster is the sum of distances from points to the cluster representative - this is the standard k-median-type cost function). One attractive feature of this formulation is that it does not require us to specify the number of clusters, or in fact any parameter.

Database Repair. In a database, *integrity constraints* like functional dependencies are used to enforce semantic consistency of the data. A simple example of a functional dependency is a *key*: if two tuples have the same value for a key attribute, they must be identical in all attributes. For any database and set of such integrity constraints, a *conflict* is a pair of tuples that does not satisfy the integrity constraint[1].

When a database with conflicts is encountered, one approach to rectify the problem is to *repair* the database by changing tuples to eliminate conflicts. For example if a database had tuples $a = (1, 2)$ and $b = (1, 3)$, but the first attribute was required to be a key, then the database could be repaired either by modifying the first component of a or the second component of b. But each of these possibilities incurs a cost that we would want to keep as low as possible.

Now if we construct a graph where each tuple is a vertex, there is an edge between two tuples if they are in conflict with respect to integrity constraints, and a distance function d captures the cost of changing one tuple to another, the problem of database repair is precisely MWIS. This problem was first formulated by Kolahi et al.[18] and NP-hardness was established via its generalization of VERTEX COVER.

1.1 Our Contributions

In this paper, we study MWIS in its many forms. Our results reveal that the interaction between the conflict graph and the distance measure plays an important role in determining the complexity of the problem. Because of the problem's similarity to clustering (and k-median in particular) one might expect that techniques that have been used to deal with k-median and related problems might be useful in attacking MWIS and potentially give a good approximation algorithm. Our first result shows that this is, in fact, impossible. More formally, we show that there is no polynomial time approximation algorithm that guarantees the approximation ratio of $O(\frac{d_{\max}}{d_{\min}})$, unless $\mathsf{P} = \mathsf{NP}$. This matches the approximation ratio given by the vertex cover heuristic up to a constant factor.

Our proof uses the fact that the graph G and metric d can be completely unrelated to each other, and this suggests that establishing a relationship between G and d could make the problem easier. We have two ways to establish consistency between d and G.

- **Metric induced by graph (w-MWIS):** Given graph $G = (V, E)$ and weights $w : E \to \mathbb{R}$, the metric d is defined as the shortest path metric of (G, w). We denote this problem by w-MWIS.
- **Graph induced by metric (d-MWIS):** Given a metric d, the conflict graph G is defined to be the induced unit-disk graph: $E(G) = \{uv : d(u, v) \leq 1\}$. This problem is abbreviated by d-MWIS.

[1] In general, many different kinds of integrity constraints (but not all!) admit pairs of conflicting tuples.

Table 1 Summary of Our Results. All hardness results are proved under the assumption of P ≠ NP. VC refers to VERTEX COVER.

	w-MWIS				d-MWIS	
Bound	General	Unweighted	Tree-Width	Planar	General	$\mathbb{R}^d$
Upper	$\frac{2d_{\max}}{d_{\min}}$ [18]	VC[18]	Poly	??	$\frac{2d_{\max}}{d_{\min}}$ [18]	$(O(1/\epsilon), 1-\epsilon)$
Lower	$\Omega(\frac{d_{\max}}{d_{\min}})$	VC	Poly		NP-hard	$\Omega(\frac{d_{\max}}{d_{\min}})$

It turns out that d-MWIS captures the class of instances that usually arise in practice, especially when d has low dimensional structure (e.g. Euclidean space or metric of bounded doubling dimension), so algorithmic results for d-MWIS are interesting from a practical point of view.

d-MWIS is hard to approximate in a general metric. Therefore we consider the d-MWIS problem in $\mathbb{R}^d$ and prove that it is NP-hard even when $d = 2$ (and even in the bi-criteria setting). We also show a constant factor bi-criteria approximation algorithm, in the following sense: If C is the minimum possible clustering cost which guarantees that centers are of distance at least 1 apart, our algorithm always outputs a solution of cost $O(C/\epsilon)$ while ensuring that the centers are "almost feasible", i.e. the distance between each pair is at least $(1-\epsilon)$. Our algorithm combines tools from clustering and computational geometry.

Similarly, w-MWIS remains hard in general weighted graphs and so we further investigate two directions. In the first direction, we consider the unweighted graph, i.e. $w(e) = 1$ for all $e \in E(G)$, and observe that the problem is computationally equivalent to VERTEX COVER, so it is hard to approximate to within a factor of $(2-\epsilon)$ assuming UGC [17], 1.36 hard assuming P≠NP [10], and admits slightly better than 2 approximation [14, 16]. In the second direction, we restrict the underlying graph structure, showing that in graphs having bounded treewidth, d-MWIS is polynomial time solvable, while it is still NP-hard in planar graphs.

Finally, a special case of MWIS captures the k-median problem which has received a lot of attention in approximation algorithms [20, 8, 1]. That is, when graph G is simply a union of disjoint cliques, the problem generalizes k-median problem and is a special case of matroid median problem, a more general variant of the classical k-median problem [9, 13, 19]. We summarize these results in Table 1.

Organization. We show our hardness results in Section 3. Results related to d-MWIS and w-MWIS are presented in Section 4 and 5 respectively. The connection between MWIS and the classical k-median problem is deferred to the full version.

2 Preliminaries

Vertex Cover: Given a graph $G = (V, E)$, a set $C \subseteq V$ is a *vertex cover* of G if any edge $uv \in E$ satisfies $\{u, v\} \cap C \neq \emptyset$. The MINIMUM VERTEX COVER problem asks for the minimum cardinality vertex cover of G. A set $S \subseteq V$ is an *independent set* of G if any pair $u, v \in S$ satisfies $uv \notin E(G)$. S is an independent set iff $V \setminus S$ is a vertex cover.

Tree Decomposition: A tree decomposition of graph G is a tree T and collection of subsets of nodes $\mathcal{X} = \{X_t\}_{t \in V(T)}$ with the following properties:

- $V(G) = \bigcup_{t \in V(T)} X_t$.
- For any edge $uv \in E(G)$, there exists $t \in V(T)$ such that both u and v lie in X_t.
- For any $v \in V(G)$, if $v \in X_t \cap X_{t'}$ for $t, t' \in V(T)$, then $v \in X_{t''}$ for any t'' that lies on the unique path from t to t' in T.

The *width* of a tree decomposition is denoted by $\max_{t \in V(T)} (|X_t| - 1)$, and the treewidth of a graph is the minimum possible k such that there is a tree decomposition of width k for G. Bodlaender [4] presented a linear time algorithm that constructs a tree decomposition of width k if the treewidth of a graph is at most k.

Bi-criteria Approximation: Let $\beta < 1$. We say that an algorithm for d-MWIS is an (α, β) approximation if it returns the solution S of open facilities whose cost is at most OPT, i.e. $\sum_{v \in V} d(S, v) \leq \alpha \mathsf{OPT}$ and for each pair $u, v \in S$, we have $d(u, v) \geq \beta$.

3 Hardness of Approximation

In this section, we prove the following theorem.

▶ **Theorem 1.** *Unless $P = NP$, d-MWIS and w-MWIS have no polynomial time $O(\Delta/\delta)$ approximation algorithms, where $\Delta = \max d(x, y)$ and $\delta = \min d(x, y)$.*

The reduction starts from the hardness of approximation result for 3SAT. We will use the following tight result due to Hastad.

▶ **Theorem 2** ([15])**.** *Fix any $\epsilon > 0$ and assume that $P \neq NP$. Given a 3SAT formula, it is hard to distinguish between the following two cases in polynomial time:*

- *(Yes-Instance:) There is an assignment that satisfies every clause.*
- *(No-Instance:) Any assignment satisfies at most $7/8 + \epsilon$ fraction of the clauses.*

Construction: We will construct an instance of w-MWIS and argue later that it can be thought of as an instance of d-MWIS as well. We use a reduction idea introduced by Feige et al. [12]. This reduction has been a powerful tool in transforming any constraint satisfaction problem (CSP) into graph G such that the size of maximum independent set of the graph is "proportional" to the value of the input CSP, i.e. the maximum fraction of clauses that can be simultaneously satisfied.

Consider an instance Φ of 3SAT with n variables $\{x_1, \ldots, x_n\}$ and m clauses $C_1, \ldots, C_m$, as given by the above theorem. We show how to construct graph $G = (V, E)$. For each clause C_i, we create a set V_i containing 7 vertices, where each such vertex represents a satisfying assignment for clause C_i. Now vertex set V can be defined as $V = \bigcup_{i=1}^{m} V_i$, so $|V| = 7m$.

We proceed to define the set of edges E. We connect every pair of vertices in V_i by an edge, so each set V_i is a clique of 7 vertices. The distance between two vertices $u, v \in V_i$ is defined as $d(u, v) = 1$. Then we say that any two vertices $u \in V_i, v \in V_j$ where $i \neq j$ are *conflicting* if there is a variable x_ℓ in the formula Φ that belongs to both clauses C_i and C_j such that the assignment of u and v set $x_\ell = 0$ and $x_\ell = 1$ respectively. Then for any two conflicting vertices $u \in V_i$ and $v \in V_j$, we connect an edge uv with distance $d(u, v) = \rho$ where ρ is a parameter. It is easy to verify that d is indeed a metric.

Analysis: It can be argued that in the Yes-Instance, there is a solution of cost at most $6m$, while in the No-Instance the cost must be at least $7m\rho/10$. So we get a gap of $\Omega(\rho)$. We omit details in this extended abstract.

4 Graph Induced by Metric (d-MWIS)

The main result in this section is a $(1/\epsilon, 1 - \epsilon)$ bi-criteria approximation algorithm for Euclidean space. The idea of this algorithm follows the LP rounding algorithm for k-median

of Charikar et al [8]. The "natural" LP requires independent set constraints, and these yield an unbounded integrality gap even for the line metric. Hence, we need a stronger LP.

First, we argue that the problem already becomes non-trivial in $\mathbb{R}^2$ even if we only desire a bi-criteria approximation algorithms.

▶ **Theorem 3.** *d-MWIS is NP-hard even in $\mathbb{R}^2$. Moreover, finding $(1, 1-\epsilon)$ approximation is also NP-hard for any $\epsilon < 1/2$.*

The proof follows by showing a reduction from VERTEX COVER on coin graphs (which is NP-hard [5])and is deferred to the full version. We also note that the problem in 1-dimensional Euclidean space is polynomial time solvable by a simple dynamic program.

4.1 Geometric LP Relaxation

Since the independent set constraints cause large LP integrality gap, we use the LP with clique point constraints, as used successfully in many geometric packing problems (see, e.g., [11, 7, 6]). We have variables $x(u)$ to indicate the fact that the facility is open at u and variables $y(u, v)$ which say that the client at v is served by a facility at u. We first formulate the linear program, denoted by (LP).

(LP)

$$\min \sum_{u,v \in V(G)} d(u,v)y(u,v)$$

$$\text{s.t} \sum_{u:d(u,p)<1/2} x(u) \leq 1 \ (\forall p \in \mathbb{R}^2)$$

$$y(u,v) \leq x(u) \ \forall u, v \in V(G)$$

$$\sum_{v \in V(G)} y(u,v) \geq 1 \ \forall u \in V(G)$$

(LP')

$$\min \sum_{u,v \in V(G)} d(u,v)y(u,v)$$

$$\text{s.t.} \sum_{u:p \in r_u} x(u) \leq 1 \ (\forall p \in \mathcal{P})$$

$$y(u,v) \leq x(u) \ \forall u, v \in V(G)$$

$$\sum_{v \in V(G)} y(u,v) \geq 1 \ \forall u \in V(G)$$

The first set of constraints ensure that we do not open conflicting centers, while the second set of constraints force the LP to only assign clients to facilities that are open. Notice an important property of this LP that, once vector $\mathbf{x}$ is fixed, the values of $\mathbf{y}$ that minimize the LP cost can be computed efficiently, so we will sometimes refer to any LP solution as $\mathbf{x}$, instead of $(\mathbf{x}, \mathbf{y})$. The following lemma says that this LP is a valid formulation for MWIS.

▶ **Lemma 4.** *Any integral vector $\mathbf{x}$ is feasible for (LP) if and only if the set $S = \{u : x(u) = 1\}$ is a feasible solution to d-MWIS.*

Proof. First, assume that $\mathbf{x}$ is integral but not feasible for (LP), so it must violate some constraint $\sum_{p:d(u,p)<1/2} x(u) > 1$ for some point p. We will argue that the solution S is not feasible. Since $\mathbf{x}$ is integral, there must be at least two vertices u, u' such that $x(u) = x(u') = 1$ and that $d(u, p), d(u', p) < 1/2$, implying that $d(u, u') \leq d(u, p) + d(p, u') < 1$. Since $u, u' \in S$, the set S is not feasible for MWIS.

Now assume that the set S defined this way is not feasible for MWIS. We will argue that $\mathbf{x}$ violates some constraint of (LP). Since S is not feasible, there must be two facilities $u, u' \in S$ such that $d(u, u') < 1$. We pick the point p to be at the middle of the line segment connecting u and u', so we must have $d(u, p), d(u', p) < 1/2$, so the LP constraint is violated at point p. ◀

From now on, we think of each vertex $u \in V(G)$ as a geometric object, i.e. a ball r_u of radius $1/2$, and therefore the first set of LP constraints can also be seen as $\sum_{u:p \in r_u} x(u) \leq 1$.

If we consider the arrangement formed by the balls, then in each cell of this arrangement, for each point p in the cell, the sum $\sum_{d(u,p)<1/2} x(u)$ is a fixed constant. Hence, we need only choose one representative point in each cell to verify the constraints. Since the arrangement of balls in d dimensions has complexity $n^{O(d)}$ (by using a simple lifting map argument) and can be computed in similar time, we can write a new polynomial sized LP, denoted by (LP').

▶ **Lemma 5.** *Vector* $\mathbf{x}$ *is feasible for (LP) if and only if it is feasible for (LP').*

Proof. We only need to show that a feasible solution $\mathbf{x}$ for (LP') is always feasible for (LP). Suppose not. Then there must be a point $p \in \mathcal{P}$ such that $\sum_{u:p\in r_u} x(u) > 1$, and since the set of r_u containing p is maximal, there is a point $p' \in \mathcal{P}$ such that the same collection of disks contains p'. Hence, the constraint of (LP') is violated at p', a contradiction. ◀

4.2 LP Rounding Algorithm

Let ϵ be a parameter. We present $(O(1/\epsilon), 1-\epsilon)$ approximation algorithm. In the first step, we perform the clustering process as used in Charikar et al.[9] where clients are grouped into many clusters. Each cluster contains, roughly speaking, clients that will be served by the same facility. In the second step, we open facilities in some of these clusters. The property of the geometric LP guarantees that the opened clusters are "far".

Step 1: Clustering and Preprocessing. We define for each vertex $v \in V(G)$, the quantity $\bar{C}_v = \sum_u d(u,v)y(u,v)$. The term $\bar{C}_v$ represents the "fractional connecting cost" of v, and we can write $\mathsf{OPT} = \sum_{v\in V(G)} \bar{C}_v$. So our goal is to use this term to bound the actual cost created by our algorithm.

First, we say that a vertex $v \in V(G)$ is *heavy* if $\bar{C}_v \geq \epsilon/10$; otherwise, the vertex v is called *light*. Denote the sets of heavy and light vertices by V^h and V^l respectively. So we can partition vertices in $V(G)$ into $V(G) = V^h \cup V^l$, which will be handled separately by our algorithm: To deal with light vertices, we order the vertices in V^l by their values $\bar{C}_v$ in increasing order, i.e. $V^l = \{v_1, v_2, \ldots, v_{n'}\}$ such that $\bar{C}_{v_1} \leq \bar{C}_{v_2} \leq \ldots \leq \bar{C}_{v_{n'}}$. Initially, we define $\mathcal{C} = \emptyset$. We process the vertices in this order, starting from v_1. When v_i is processed, we consider $V_i = \left\{v_{i'} \in V^l : d(v_i, v_{i'}) \leq 3\bar{C}_{v_i}\right\}$ and check if there is another $j : j < i$ such that $v_j \in \mathcal{C}$ and $V_i \cap V_j \neq \emptyset$. If there is no such cluster, we add v_i to the collection of cluster centers $\mathcal{C}$; otherwise, we say that vertex v_i is *assigned* to center v_j.

When the above process finishes, we obtain a collection of cluster centers $\mathcal{C}$. The following observations follow immediately.

▶ Observation 4.1. For any vertex $v_i \in V^l$, either $v_i \in \mathcal{C}$ or vertex v_i is assigned to some other vertex $v_j \in \mathcal{C}$. In the second case, $d(v_i, v_j) \leq 3\bar{C}_{v_i} + 3\bar{C}_{v_j}$.

▶ Observation 4.2. For any $V_i, V_j \in \mathcal{V}$, we have $V_i \cap V_j = \emptyset$.

For each cluster center $v_j \in \mathcal{C}$, we define the combined demand D_j to be the number of vertices assigned to v_j, including v_j itself. Observe that $\sum_{j\in\mathcal{C}} D_j = |V^l|$. From now on, we will think of v_j as the representative of all clients assigned to it, and we will only try to open facilities to serve v_j. Now our goal is to solve the following new problem, instead of the original one: Given a collection of clients $\mathcal{C}$, the goal is to open the non-conflicting centers among the vertices in $\mathcal{C}$ so as to minimize the demand-weighted cost $\sum_{v_j\in\mathcal{C}} D_j d(S, v_j)$.

We ensure that the solution of this new problem can be turned into that of the old one without paying much cost, as stated in the following lemma.

▶ **Lemma 6.** *Let $S \subseteq \mathcal{C} \cup V^h$ be any subset of vertices that is maximal, w.r.t. V^h in the sense that any vertex $u \in V^h$ either belongs to S or is in conflict with some other vertex $u' \in S$. Then we have $\sum_{v \in V(G)} d(S,v) \leq O(1/\epsilon)\mathsf{OPT} + \sum_{v_j \in \mathcal{C}} D_j d(S, v_j)$*

Proof. Recall that $V(G) = V^l \cup V^h$. Since S is maximal, we know that the connecting cost of any vertex is never more than one. We first analyze the connecting cost of vertices in V^h. Consider vertex $v \in V^h$. Since $\bar{C}_v \geq \epsilon/10$, we have $d(S,v) \leq 1 \leq 10\bar{C}_v/\epsilon$. Summing over all $v \in V^h$, we get $\sum_{v \in V^h} d(S,v) \leq 10 \sum_{v \in V^h} \bar{C}_v/\epsilon \leq (10/\epsilon)\mathsf{OPT}$.

Now we bound the connecting cost of vertices v_i in V^l. If vertex v_i is the center of some cluster, i.e. $v_i \in \mathcal{C}$, we have the connecting cost $d(v_i, S)$. Otherwise from Observation 4.1, v_i is assigned to some cluster center $v_j \in \mathcal{C}, j < i$, so the connecting cost from v_i to the nearest opened facility is at most $d(S,v_i) \leq d(v_i,v_j) + d(S,v_j)$. Since $j < i$, it must be the case that $\bar{C}_{v_i} \geq \bar{C}_{v_j}$, so we have $d(v_i,v_j) \leq 3\bar{C}_{v_i} + 3\bar{C}_{v_j} \leq 6\bar{C}_{v_i}$. Finally, we have $d(S,v_i) \leq 6\bar{C}_{v_i} + d(S,v_j)$ in this case, so if we sum over all vertices v_i that have been assigned to v_j, we would obtain the bound $6\sum_{i:v_i \text{ assigned to } v_j} \bar{C}_{v_i} + D_j d(S,v_j)$.

Summing over all vertices gives $\sum_{v \in V^l} d(v,S) \leq 6\sum_{v \in V^l} \bar{C}_v + \sum_{v_j \in \mathcal{C}} D_j d(S,v_j)$. ◀

Step 2: Opening the Facilities. For each $v_j \in \mathcal{C}$, we simply open facility v_j. Then, we process heavy vertices in V^h in arbitrary order, and we open a facility if it does not conflict with any other already opened facility (remark that the conflicts might have been created already by opening all vertices $v_j \in \mathcal{C}$, but we avoid creating more conflict). This algorithm ensures that the resulting set S of opened facilities is a maximal set with respect to V^h, and observe that $\sum_{v_j \in \mathcal{C}} d(S,v_j) = 0$. So invoking Lemma 6 implies that $\sum_{v \in V(G)} d(S,v) \leq O(\mathsf{OPT}/\epsilon)$.

It only remains to analyze the distance between centers in S, which is done in the following lemma.

▶ **Lemma 7.** *Let S be the set of facilities opened by the algorithm. Then, for any pair $u, v \in S$, $d(u,v) \geq 1 - \epsilon$.*

Proof. Notice that we only need to analyze the pair of vertices in S that were once the clusters in $\mathcal{C}$ (because other vertices are added arbitrarily in a way that ensure no conflict). We will need the following claim.

▶ Claim 4.1. For all $v_i \in \mathcal{C}$, we have $\sum_{u \in V_i} x(u) \geq 2/3$

Proof. In fact, this claim can be seen as just a simple application of Markov's inequality: The term $\bar{C}_{v_i}$ is simply an expectation of connecting cost of v_i, where v_i is connected to u with probability $x(u)$. To be more formal, we write $\bar{C}_{v_i} = \sum_v y(v,v_i) d(v,v_i)$. Since $\sum_v y(v,v_i) = 1$, the terms $\{y(v,v_i)_v\}$ can be seen as distribution μ such that $\mathbb{E}_{v \sim \mu}[d(v,v_i)] = \bar{C}_{v_i}$. Applying Markov's inequality, the probability that $d(v,v_i) > 3\bar{C}_{v_i}$ is at most $1/3$, or in other words, $\sum_{v:d(v,v_i) \leq 3\bar{C}_{v_i}} y(v,v_i) \geq 2/3$. This implies that $\sum_{v:d(v,v_i) \leq 3\bar{C}_{v_i}} x(v) \geq 2/3$, as desired. ◀

Now consider $v_i, v_j \in \mathcal{C}$ and assume (for contradiction) that $d(v_i,v_j) < 1 - \epsilon$. Consider a point p whose distance to v_i and v_j are equal, so we have $d(p,v_i) = d(p,v_j) < 1/2 - \epsilon/2$. By triangle inequality, for any $u \in V_i$, the distance $d(u,p) \leq d(u,v_i) + d(v_i,p) \leq 1/2 - \epsilon/2 + 3\epsilon/10 < 1/2$ (also because $d(u,v_i) \leq 3\bar{C}_{v_i} \leq 3\epsilon/10$.)

So the balls r_u contain point p for all $u \in V_i$. By similar arguments, balls r_v contain point p for all $v \in V_j$, and this implies that $\sum_{u:p \in r_u} x(u) \geq \sum_{u \in V_i} x(u) + \sum_{u \in V_j} x(u) \geq 4/3$, from the claim and Observation 4.2; this contradicts the fact that the LP is feasible. Notice that point p may not belong to $\mathcal{P}$, but from the fact that $\mathcal{P}$ contains important points, we must have another point $p' \in \mathcal{P}$ such that $\sum_{u:p' \in r_u} x(u) > 4/3$. ◀

5 Metric Induced by Graphs (w-MWIS)

Recall that in the w-MWIS problem, we are given a graph G together with weight function $w : E(G) \to \mathbb{R}_{\geq 0}$. Metric d is then defined as an induced shortest path metric on (G, w). In this section, we consider special cases of w-MWIS when $w = 1$ and when the treewidth of graph G is bounded.

5.1 Unweighted Graphs

In this section, we consider the metric induced by unweighted graph (i.e. $w(e) = 1$ for all $e \in E(G)$) and prove that the problem is again equivalent to VERTEX COVER; the proof is simple and deferred to the full version.

▶ **Theorem 8.** *Unweighted w-MWIS is computationally equivalent to Vertex Cover. More formally, for any ρ, there is an algorithm that gives a ρ-approximation to w-MWIS on G if and only if there is a ρ-approximation to Vertex Cover G.*

One corollary of the above theorem is that, w-MWIS becomes NP-hard even in planar graphs, and there is a PTAS when the graph is unweighted (this follows from the fact that the minimum vertex cover problem admits a PTAS in planar graphs [2]).

▶ **Corollary 9.** MWIS *is NP-hard on unweighted planar graphs.*

5.2 Algorithm for weighted trees

In this section, we consider d-MWIS on trees and give a polynomial time algorithm for computing the optimal solution. Let (T, w) be the input where T is a tree and $w : E(T) \to \mathbb{R}$ is a weight function. Recall that VERTEX COVER is solvable in polynomial time on trees, and since VERTEX COVER is equivalent to w-MWIS when $w = 1$, our algorithm can be thought of as a strengthened version of vertex cover algorithm to handle a more general problem.

First we show some structural properties that suggest our dynamic programming. The solution for MWIS can be described by a set S of centers that are open, and given such set S, we can naturally define an assignment function $\alpha : V(G) \to S$ that assigns each vertex $v \in V(G)$ to its closest opened center. Let T denote the tree instance with the corresponding induced metric $d_T : V(T) \times V(T) \to \mathbb{R}$. We root the tree at arbitrary vertex $r \in V(T)$, and define, for each vertex $v \in V(T)$, the subtree T_v as the subtree of T rooted at v (including v itself). The following lemma characterizes the properties of the optimal solution on the tree.

▶ **Lemma 10.** *Let S^* be the set of opened centers in the optimal solution and $\alpha^* : V(T) \to S^*$ be the corresponding assignment. The following properties hold:*

- *For two vertices $u, v \in V(T)$ such that v is a parent of u in the tree, if $\alpha^*(v) \in T_u$ then $\alpha^*(u) = \alpha^*(v)$.*
- *For any two vertices u, v, we have $d_T(\alpha^*(u), u) \leq d_T(\alpha^*(v), u)$*

Proof. To prove the first property, assume this is not the case. Then there must be another vertex $w \in T_v \cap S^*$ such that $d_T(w, u) < d_T(\alpha^*(v), u)$, but this implies that $d_T(w, v) = d_T(w, u) + d_T(u, v) < d_T(\alpha^*(v), u) + d_T(u, v) = d_T(\alpha^*(v), v)$, a contradiction. The second property is obvious from the definition of α^*. ◀

Algorithm

Define the subproblem $C(v, x)$ as the minimum possible assignment cost of vertices in the subtree T_v with the constraint that vertex v is assigned to an opened center x (which may not necessarily belong to T_v). The optimal solution we are looking for is stored in the entry $C(r, \alpha^*(r))$ (which can be enumerated once the entries are computed correctly). The table entries are computed from the leaf to the root of the tree. The base case is defined on the leaves of the tree simply by: For each leaf $v \in V(T)$, we define $C(v, x) = d_T(v, x)$.

Now fix an entry $C(v, x)$. We show how to compute $C(v, x)$ once the subproblems have been computed. Let $v_1, \ldots, v_\ell$ be the children of v in the tree. For each child v_i such that $x \notin T_{v_i}$, we define the set of possible center candidates for v_i as

$$\Gamma_{v_i,x} = \{x\} \cup \{y \in T_{v_i} : yx \notin E(T), d_T(v_i, y) \leq d_T(v_i, x) \text{ and } d_T(v, y) \geq d_T(v, x)\}$$

Otherwise, if $x \in T_{v_i}$, define $\Gamma_{v_i,x} = \{x\}$. Then we can write our recurrence as:

$$C(v, x) = d_T(v, x) + \sum_{i=1}^{\ell} \min_{y \in \Gamma_{v_i,x}} C(v_i, y)$$

It follows from Lemma 10 that restricting our choices to $\Gamma_{v_i,x}$ still includes the optimal solution, so we only need to show that a feasible solution can be constructed from the table entries.

Correctness

We show that given the table entries $C(v, x)$ for all $v, x \in V(T)$, we can reconstruct a feasible solution that assigns vertices in T_v with total cost $C(v, x)$. This can be argued by induction on the tree structure as summarized in the following lemma.

▶ **Lemma 11.** *For each table entry $C(v, x)$, we can construct a corresponding partial solution $S^*_{(v,x)}$ (where $S^*_{(v,x)}$ is the set of opened centers) such that the assignment cost inside the subtree T_v is bounded by $\sum_{u \in T_v} d_T(S^*_{(v,x)}, u) \leq C(v, x)$. Moreover $S^*_{(v,x)} \subseteq T_v \cup \{x\}$, $x \in S^*_{(v,x)}$, and $S^*_{(v,x)}$ is an independent set in T.*

Proof. We will prove this lemma by induction on the distance of vertices from the leafs (i.e. in order from leafs to root). The base case when v is a leaf is trivial: We can simply define $S^*_{(v,x)} = \{x\}$ for all $x \in V(T)$, so this is clearly an independent set where $x \in S^*_{(v,x)}$ and $S^*_{(v,x)} \subseteq T_v \cup \{x\}$. Now consider a vertex v with children $v_1, \ldots, v_\ell$ and assume that the induction hypothesis holds for $C(v_i, y)$ for any v_i and y. Our goal is to show that it holds for $C(v, x)$.

For each $i \in [\ell]$, let $y_i \in \Gamma_{v_i,x}$ be the vertex such that $C(v_i, y_i)$ is minimized, so we can write $C(v, x)$ as $C(v, x) = d_T(v, x) + \sum_{i=1}^{\ell} C(v_i, y_i)$. We construct solution $S^*_{(v,x)}$ by $S^*_{(v,x)} = \{x\} \cup \bigcup_{i=1}^{\ell} S^*_{(v_i,y_i)}$, so it is immediate that $x \in S^*_{(v,x)}$. It only remains to show that (i) $S^*_{(v,x)}$ is an independent set, (ii) $S^*_{(v,x)} \subseteq T_v \cup \{x\}$, and (iii) that $\sum_{u \in T_v} d_T(S^*_{(v,s)}, u) \leq C(v, x)$.

To prove (i), assume (for contradiction) that there is an edge $w_i w_j$ for $w_i \in S^*_{(v_i,y_i)}$ and $w_j \in S^*_{(v_j,y_j)}$ for some $i \neq j$. If both $w_i \in T_{v_i}$ and $w_j \in T_{v_j}$, then it is impossible to have an edge between them, so there must be one of them that lies outside its tree. Assume that $w_i \notin T_{v_i}$ (the other case is symmetric), we must have $w_i = y_i$ because $S^*_{(v_i,y_i)} \subseteq T_{v_i} \cup \{y_i\}$. And since $y_i \in \Gamma_{v_i,x} \subseteq T_{v_i} \cup \{x\}$, the only possibility for y_i to be outside of T_{v_i} is that $y_i = x$. We now do case analysis.

- If $w_i \notin T_{v_i}$ but $w_j \in T_{v_j}$, either ($w_i = v$ and $w_j = v_j$) or $w_i \in T_{v_j}$. The latter is impossible because it would force $\Gamma_{v_j,x} = \{x\}$ (because $x \in T_{v_j}$), and hence both $w_i = x$ and w_j belong to $S^*_{(v_j,y_j)}$ by induction hypothesis, contradicting the fact that $S^*_{(v_j,y_j)}$ is independent. For the former case, $w_j = v_j$ implies that $w_j \notin \Gamma_{v_j,x}$, and in particular $w_j \neq y_j$. Now since $S^*_{(v_j,y_j)}$ is computed from $\{y_j\} \cup \bigcup_{v'_r} S^*_{(v'_r,y'_r)}$ where vertices v'_r are children of v_j, we must have w_j in some set $S^*_{(v'_r,y'_r)}$. This is only possible if $w_j = y'_r$ for some r, a contradiction to the fact that $w_j \notin \Gamma_{v'_r,y_j}$.
- If $w_i \notin T_{v_i}$ and $w_j \notin T_{v_j}$, then $w_i = y_i$ and $w_j = y_j$, but since $y_i \in T_{v_i} \cup \{x\}$, we must have $w_i = x$; similarly, since $y_j \in T_{v_j} \cup \{x\}$, it must be the case that $w_j = x$, a contradiction.

Next, we turn to prove (ii). Assume for contradiction that $S^*_{(v,x)} \not\subseteq T_v \cup \{x\}$. Denote by $z \in S^*_{(v,x)} \setminus (T_v \cup \{x\})$. Since $S^*_{(v,x)}$ is obtained from the union of $S^*_{(v_i,y_i)}$ and $\{x\}$, it must be the case that $z \in S^*_{(v_i,y_i)}$ for some i, and since y_i is the only node outside of T_{v_i}, we have $z = y_i \neq x$. This implies that $y_i \notin \Gamma_{i,x}$, a contradiction.

Finally, to prove (iii), we write $C(v,x)$ as $C(v,x) = d_T(v,x) + \sum_{i=1}^{\ell} C(v_i,y_i)$, where $C(v_i,y_i) \geq \sum_{u \in T_{v_i}} d_T(S^*_{(v_i,y_i)},u)$ by induction hypothesis. Since $x \in S^*_{(v,x)}$, we have that $d_T(S^*_{(v,x)},v) \leq d_T(v,x)$. Moreover, for each $u \in T_{v_i}$, $d_T(S^*_{(v,x)},u) \leq d_T(S^*_{(v_i,y_i)},u)$ because $S^*_{v_i} \subseteq S^*_v$, and $\sum_{u \in T_{v_i}} d_T(S^*_{(v_i,y_i)},u) \leq C(v_i,y_i)$ by induction hypothesis. This implies

$$\sum_{u \in T_v} d_T(S^*_{(v,x)},u) = d_T(S^*_{(v,x)},v) + \sum_i \sum_{u \in T_{v_i}} d_T(S^*_{(v,x)},u) \leq d_T(v,x) + \sum_{i=1}^{\ell} C(v_i,y_i) = C(v,x)$$

◀

5.3 Extension to Graphs of Bounded Treewidth

In this section, we extend the algorithm on trees to give polynomial time algorithm for graphs of bounded treewidth. The key ideas remain the same as the tree case, but the algorithm and analysis are more involved. For a subset $S \subseteq V(G)$, denote by $G[S]$ an induced subgraph of G on vertices in S. Let $(T,\mathcal{X})$ be a tree decomposition of G and assume that the tree is rooted at an arbitrary vertex $r \in T$. Assume that the treewidth of G is at most w, so we have $|X_t| \leq w$ for each $t \in T$. For each vertex $t \in T$, we define $G^{(t)}$ to be an induced subgraph of G on nodes $\bigcup_{t' \in T_t} X_{t'}$, i.e. $G^{(t)} = G[\bigcup_{t' \in T_t} X_{t'}]$. These subgraphs define our subproblems.

For each $t \in V(T)$, a *feasible partial assignment* for t is a function $\gamma : X_t \to V(G)$ such that there is no edge in the induced subgraph $G[\{\gamma(v) : v \in X_t\}]$. This partial assignment is used to memorize the closest opened centers to vertices in X_t, i.e. $\gamma(v)$ is supposed to represent the node in the optimal solution S^* that is closest to v. We now state a lemma similar in spirit to Lemma 10 (with proof deferred to a full version).

▶ **Lemma 12.** *Let S^* be the set of opened centers in the optimal solution and α^* be the corresponding assignment. Let $t, t' \in V(T)$ be two vertices such that t is a parent of t'. Then the following properties hold:*

- *If $\alpha^*(v) \in G^{(t')}$ for some $v \in X_t$, then $\alpha^*(u) = \alpha^*(v)$ for some $u \in X_{t'}$.*
- *For any vertex $u \in V(G^{(t')})$, if $\alpha^*(u) \notin G^{(t')}$, then $\alpha^*(u) = \alpha^*(v)$ for some $v \in X_t$.*

Algorithm: Now we show the algorithm that solves w-MWIS optimally in time $n^{O(w)}$. The algorithm is suggested by the above lemma. For each $t \in V(T)$ and feasible partial assignment $\gamma : X_t \to V(G)$, the table entry $C(t,\gamma)$ stores the minimum possible assignment cost inside

$G^{(t)}$ such that vertex $v \in X_t$ is assigned to $\gamma(v)$ for all $v \in X_t$. The optimal solution we want is in the entry $C(r, \gamma^*)$ where γ^* is a restriction of α^* on X_r.

To compute the entry $C(t, \gamma)$, consider the children $t_1, \ldots, t_\ell$ of t in the tree. For each $i \in [\ell]$, we define the set $\Gamma_{i,\gamma}$ as the set of all possible partial assignments $\gamma' : X_{t'} \to V(G)$ that satisfy the following properties:

1. For all $v \in X_t \cap X_{t_i}$, we have $\gamma'(v) = \gamma(v)$.
2. If $\gamma(v) \in G^{(t_i)}$ for some $v \in X_t$, then $\gamma'(u) = \gamma(v)$ for some $u \in X_{t_i}$.
3. For any vertex $u \in G^{(t_i)}$, if $\gamma'(u) \notin G^{(t_i)}$, then $\gamma'(u) = \gamma(v)$ for some $v \in X_t$.
4. There is no edge $\gamma(v)\gamma'(v') \in E(G)$ for any $v, v' \in X_t \cup X_{t_i}$.

The number of possible functions γ' is at most n^w (each vertex in X_{t_i} has at most n possibilities of γ'), so this is polynomial time computable if the treewidth of G is at most a constant. Now the table entry $C(t, \gamma)$ can be computed as follows. The base case of $C(t, \gamma)$ when t is a leaf can be computed easily by $C(t, \gamma) = \sum_{v \in X_t} d(v, \gamma(v))$. Otherwise, for any t, γ such that t has $t_1, \ldots, t_\ell$ as its children, we can write the recurrence:

$$C(t, \gamma) = \sum_{i=1}^{\ell} \min_{\gamma' \in \Gamma_{i,\gamma}} C(t_i, \gamma') - \sum_{v \in X_t} (n_v - 1) d(v, \gamma(v))$$

where n_v is the number of j such that $v \in X_{t_j}$. Notice that n_v does not depend on the choices of γ.

Correctness: The proof of the following lemma uses similar ideas as in the tree case but more complicated.

▶ **Lemma 13.** *For any vertices $t \in V(T)$ and feasible partial assignment γ for t, we can construct a partial solution $S^*_{(t,\gamma)}$ such that the total assignment cost inside $G^{(t)}$ is bounded by*

$$\sum_{u \in V(G^{(t)})} d(S^*_{(t,\gamma)}, u) \leq C(t, \gamma)$$

Moreover, $S^*_{(t,\gamma)} \subseteq V(G^{(t)}) \cup \{\gamma(u) : u \in X_t\}$, $\gamma(X_t) \subseteq S^*_{(t,\gamma)}$ and $S^*_{(t,\gamma)}$ is an independent set in G.

Acknowledgement. We thank the FSTTCS reviewers for many useful comments, espcially for suggesting a simplified connection between our problem and the standard k-median problem.

References

1 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.

2 Brenda S. Baker. Approximation algorithms for np-complete problems on planar graphs. *J. ACM*, 41(1):153–180, January 1994.

3 Sugato Basu, Ian Davidson, and Kiri Lou Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications.* Chapman & Hall/CRC, 2009.

4 Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

5 M.R. Cerioli, L. Faria, T.O. Ferreira, and F. Protti. On minimum clique partition and maximum independent set on unit disk graphs and penny graphs: complexity and approximation. *Electronic Notes in Discrete Mathematics*, 18(0):73 – 79, 2004.

6 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In Claire Mathieu, editor, *SODA*, pages 892–901. SIAM, 2009.

7 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012.

8 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *J. Comput. Syst. Sci.*, 65(1):129–149, 2002.

9 Moses Charikar and Shi Li. A dependent lp-rounding approach for the k-median problem. In *Automata, Languages, and Programming*, pages 194–205. Springer, 2012.

10 Irit Dinur and Shmuel Safra. The importance of being biased. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 33–42, New York, NY, USA, 2002. ACM.

11 Alina Ene, Sariel Har-Peled, and Benjamin Raichel. Geometric packing under non-uniform constraints. In Tamal K. Dey and Sue Whitesides, editors, *Symposium on Computational Geometry*, pages 11–20. ACM, 2012.

12 Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *J. ACM*, 43(2):268–292, 1996.

13 MohammadTaghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Local search algorithms for the red-blue median problem. *Algorithmica*, 63(4):795–814, 2012.

14 Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM J. Comput.*, 31(5):1608–1623, 2002.

15 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, July 2001.

16 George Karakostas. A better approximation ratio for the vertex cover problem. *ACM Transactions on Algorithms*, 5(4), 2009.

17 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.*, 74(3):335–349, 2008.

18 Solmaz Kolahi, Laks V. S. Lakshmanan, Jonathan Leung, Divesh Srivastava, and Suresh Venkatasubramanian. Data cleaning 2.0: Scalable generation of association-preserving repairs. Manuscript., 2013.

19 Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. The matroid median problem. In Dana Randall, editor, *SODA*, pages 1117–1130. SIAM, 2011.

20 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 901–910. ACM, 2013.