Individual Preference Stability for Clustering

Saba Ahmadi [1] **Pranjal Awasthi** [2] **Samir Khuller** [3] **Matthäus Kleindessner** [4] **Jamie Morgenstern** [5]
Pattara Sukprasert [3] **Ali Vakilian** [1]

Abstract

In this paper, we propose a natural notion of individual preference (IP) stability for clustering, which asks that every data point, on average, is closer to the points in its own cluster than to the points in any other cluster. Our notion can be motivated from several perspectives, including game theory and algorithmic fairness. We study several questions related to our proposed notion. We first show that deciding whether a given data set allows for an IP-stable clustering in general is NP-hard. As a result, we explore the design of efficient algorithms for finding IP-stable clusterings in some restricted metric spaces. We present a polytime algorithm to find a clustering satisfying exact IP-stability on the real line, and an efficient algorithm to find an IP-stable 2-clustering for a tree metric. We also consider relaxing the stability constraint, i.e., every data point should not be too far from its own cluster compared to any other cluster. For this case, we provide polytime algorithms with different guarantees. We evaluate some of our algorithms and several standard clustering approaches on real data sets.

1. Introduction

Clustering, a foundational topic in unsupervised learning, aims to find a partition of a dataset into sets where the intra-set similarity is higher than the inter-set similarity. The problem of clustering can be formalized in numerous ways with different ways of measuring similarity within and between sets, such as centroid-based formulations like k-means, k-median and k-center (e.g., Arthur and Vassilvitskii, 2007), hierarchical partitionings (e.g., Dasgupta, 2002), or spectral clustering (e.g., von Luxburg, 2007). All these formulations have also been considered under additional constraints (e.g., Wagstaff et al., 2001); in particular, a recent line of work studies clustering under various fairness constraints (e.g., Chierichetti et al., 2017). Most formulations of clustering are NP-hard, which has led to both the understanding of approximation algorithms and conditions under which (nearly) optimal clusterings can be found in computationally efficient ways. Such conditions that have been studied in recent years are variants of *perturbation stability*, i.e., small perturbations to the distances do not affect the optimal clustering (Ackerman and Ben-David, 2009; Awasthi et al., 2012; Bilu and Linial, 2012), or are variants of *approximation stability*, i.e., approximately optimal clusterings according to some objective are all close to each other (Balcan et al., 2013; Ostrovsky et al., 2013). In this work, we introduce a distinct notion of stability in clustering, *individual preference stability* of a clustering, which measures whether data points can reduce their individual objective by reassigning themselves to another cluster.

The study of individual preference (IP) stability, and clusterings which are IP-stable, has a number of motivations behind it, both in applications and connections to other concepts in computing. For example, clustering can be used to design curricula for a collection of students, with the goal that each student is assigned to a cluster with the most similar learning needs. One might also design personalized marketing campaigns where customers are first clustered–the campaign's personalization will be more effective if customers have most affinity to the clusters to which they are assigned. More generally, if one first partitions a dataset and then designs a collection of interventions or treatments for each subset, IP stability ensures individuals are best represented by the cluster they belong to.

This notion has connections to several other concepts studied in computing, both for clustering and for other algorithmic problems. Suppose each data point "belongs" to an individual, and that individual chooses which cluster she wants to join to minimize her average distance to points in that cluster. If her features are fixed (and she cannot change

[1]Toyota Technological Institute at Chicago, USA [2]Google, USA [3]Northwestern University, USA [4]Amazon Web Services, Germany [5]University of Washington, USA. Correspondence to: S. Ahmadi <saba@ttic.edu>, P. Awasthi <pranjalawasthi@google.com>, S. Khuller <samir.khuller@northwestern.edu>, M. Kleindessner <matkle@amazon.de>, J. Morgenstern <jamiemmt@cs.washington.edu>, P. Sukprasert <pattara@u.northwestern.edu>, A. Vakilian <vakilian@ttic.edu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

them), an IP-stable clustering will correspond to a Nash equilibrium of this game. In this sense, IP-stability is related to the concept of stability from matching theory (Roth, 1984; Gale and Sotomayor, 1985). IP-stability is also a natural notion of individual fairness for clustering, asking that each individual prefers the cluster they belong to over any other cluster (and directly captures envy-freeness of a clustering such as has been recently studied within the context of classification (Balcan et al., 2019)). Finally, the study of IP-stability (whether such a clustering exists, whether an objective-maximizing clustering is also IP-stable) may open up a new set of conditions under which approximately optimal clusterings can be found more efficiently than in the worst case.

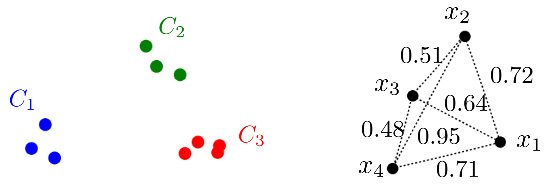

Figure 1. Two data sets in $\mathbb{R}^2$ with d equaling the Euclidean metric. **Left:** An IP-stable 3-clustering with clusters C_1, C_2, C_3. **Right:** Four points for which there is no IP-stable 2-clustering. For example, if the two clusters were $C_1 = \{x_1, x_4\}$ and $C_2 = \{x_2, x_3\}$, then x_1 would not be stable because of $d(x_1, x_4) = 0.71 > 0.68 = [d(x_1, x_2) + d(x_1, x_3)]/2$, and if the two clusters were $C_1 = \{x_1\}$ and $C_2 = \{x_2, x_3, x_4\}$, then x_4 would not be stable because $d(x_1, x_4) = 0.71 < 0.715 = [d(x_2, x_4) + d(x_3, x_4)]/2$.

Our Contributions We propose a natural notion of IP-stability clustering and present a comprehensive analysis of its properties. Our notion requires that each data point, on average, is closer to the points in its own cluster than to the points in any other cluster (Section 2). We show that, in general, IP-stable clusterings might not exist, even for Euclidean data sets in $\mathbb{R}^2$ (Section 2).

Moreover, we prove that deciding whether a given data set has an IP-stable k-clustering is NP-hard, even for $k = 2$ and when the distance function is assumed to be a metric (Section 3). Here, and in the following, k is the desired number of clusters. This naturally motivates the study of approximation algorithms for the problem and the study of the problem in special metric spaces.

A clustering is called t-approximately IP-stable if the average distance of each data point to its own cluster is not more than t times its average distance to the points in any other cluster. By exploiting the techniques from metric embedding, in Section 4.1, first we provide a polynomial time algorithm that finds an $\mathcal{O}(\log^2 n/\varepsilon)$-approximation IP-stable clustering for $(1-\varepsilon)$-fraction of points in any metric space. Second, by designing a modified single-linkage approach as a pre-processing step, in Section 4.2 we provide an "efficient" approximation algorithm when the input has a "well-separated" IP-stable clustering. More precisely, if the input has an IP-stable clustering with clusters of size at least $\alpha \cdot n$, then our algorithm will find an $\tilde{\mathcal{O}}(\frac{1}{\alpha})$-approximation IP-stable clustering of the input in polynomial time.

When the data set lies on the real line, surprisingly, we show that an IP-stable k-clustering always exists, and we design an efficient algorithm with running time $\mathcal{O}(kn)$ to find one, where n is the number of data points (Section 5.1). In addition to finding a IP-stable clustering, one might want to optimize an additional global objective; we study such a problem in Appendix F, where we minimize the deviation of clusters' sizes from desirable target sizes. We propose a DP solution which runs in $\mathcal{O}(n^3 k)$ time for this problem. Moreover, we show how to efficiently find an IP-stable 2-clustering when the distance function is a tree metric (Section 5.2). We in fact conjecture that the result on the line can be extended to any tree, and while we show a positive result only for $k = 2$, we believe that a similar result holds for any k.

Finally, we perform extensive experiments on real data sets and compare the performance of several standard clustering algorithms such as k-means++ and k-center w.r.t. IP-stability (Section 6). Although in the worst-case the violation of IP-stability by solutions produced by these algorithms can be arbitrarily large (as we show in Section 3 / Appendix E), some of these algorithms perform surprisingly well in practice. We also study simple heuristic modifications to make the standard algorithms more aligned with the notion of IP-stability, in particular for linkage clustering (Appendix G).

2. Individual Preference Stability

Our notion of individual preference stability applies to a data set $\mathcal{D}$ together with a given dissimilarity function d that measures how close two data points are. We use the terms dissimilarity and distance synonymously. We assume $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}_{\geq 0}$ to be symmetric with $d(x, x) = 0$, but not necessarily to be a metric (i.e., to additionally satisfy the triangle inequality and $d(x, y) = 0 \Leftrightarrow x = y$).

Our stability notion defines what it means that a data point is IP-stable in a clustering of $\mathcal{D}$; namely: a data point is IP-stable if the average distance to the points in its own cluster (the point itself excluded) is not greater than the average distance to the points in any other cluster. Then a clustering of $\mathcal{D}$ is said to be IP-stable if every data point in $\mathcal{D}$ is stable.

For the rest of the paper we assume $\mathcal{D}$ to be finite. Our definition of IP-stability for clustering can then be formally stated as follows (for $l \in \mathbb{N}$, we let $[l] = \{1, \ldots, l\}$):

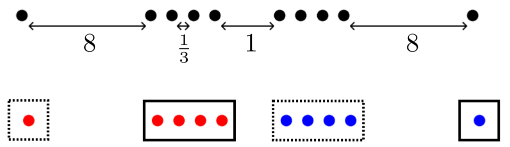

Figure 2. A data set on the real line with more than one IP-stable clustering. **Top:** The data set and the distances between the points. **Bottom:** The same data set with two IP-stable 2-clusterings (one encoded by color: red vs blue / one encoded by frames: solid vs dotted boundary).

Definition 1 (Individual preference (IP) stability)**.** *Let $\mathcal{C} = (C_1, \dots, C_k)$ be a k-clustering of $\mathcal{D}$, that is $\mathcal{D} = C_1 \dot\cup \dots \dot\cup C_k$ and $C_i \neq \emptyset$ for $i \in [k]$. For $x \in \mathcal{D}$, we write $C(x)$ for the cluster C_i that x belongs to. We say that $x \in \mathcal{D}$ is IP-stable if either $C(x) = \{x\}$ or*

$$\frac{1}{|C(x)|-1} \sum_{y \in C(x)} d(x,y) \leq \frac{1}{|C_i|} \sum_{y \in C_i} d(x,y) \qquad (1)$$

for all $i \in [k]$ with $C_i \neq C(x)$. The clustering $\mathcal{C}$ is an IP-stable k-clustering if every $x \in \mathcal{D}$ is IP-stable. For brevity, instead of IP-stable we may only say stable.

We discuss some important observations about IP-stability as defined in Definition 1: if in a clustering all clusters are well-separated and sufficiently far apart, then this clustering is IP-stable. An example of such a scenario is provided in the left part of Figure 1. Hence, at least for such simple clustering problems with an "obvious" solution, IP-stability does not conflict with the clustering goal of partitioning the data set such that "data points in the same cluster are similar to each other, and data points in different clusters are dissimilar" (Celebi and Aydin, 2016, p. 306). However, there are also data sets for which no IP-stable k-clustering exists (for a fixed k and a given distance function d).[1] This can even happen for Euclidean data sets and $k = 2$, as the right part of Figure 1 shows. If a data set allows for an IP-stable k-clustering, there might be more than one IP-stable k-clustering. An example of this is shown in Figure 2. This example also illustrates that IP-stability does not necessarily work towards the aforementioned clustering goal. Indeed, in Figure 2 the two clusters of the clustering encoded by the frames, which is stable, are not even contiguous.

These observations raise a number of questions such as: when does an IP-stable k-clustering exist? Can we efficiently decide whether an IP-stable k-clustering exists? If an IP-stable k-clustering exists, can we efficiently compute it? Can we minimize some (clustering) objective over the set of all IP-stable clusterings? How do standard clustering algorithms such as Lloyd's algorithm (aka k-means; Lloyd, 1982) or linkage clustering (e.g., Shalev-Shwartz and Ben-David, 2014, Section 22) perform in terms of IP-stability? Are there simple modifications to these algorithms in order to improve their stability? In this paper, we explore some of these questions as outlined in Section 1.

[1]Of course, the trivial 1-clustering $\mathcal{C} = (\mathcal{D})$ or the trivial $|\mathcal{D}|$-clustering that puts every data point in a singleton are stable, and for a trivial distance function $d \equiv 0$, every clustering is stable.

2.1. Related Work and Concepts

Clustering Stability There is a large body of work on the design of efficient clustering algorithms, both in the worst case and under various stability notions (Awasthi and Balcan, 2014). Some works have studied stability notions called "average stability" that are similar to our notion of IP-stability. The work of Balcan et al. (2008) studies properties of a similarity function that are sufficient in order to approximately recover (in either a list model or a tree model) an unknown ground-truth clustering. One of the weaker properties they consider is the average attraction property, which requires inequality (1) with $t = 1$ to hold for the ground-truth clustering, but with an additive gap of $\gamma > 0$ between the left and the right side of (1). Balcan et al. show that the average attraction property is sufficient to successfully cluster in the list model, but with the length of the list being exponential in $1/\gamma$, and is not sufficient to successfully cluster in the tree model. The works discussed above utilize strong forms of average stability to bypass computational barriers and recover the ground-truth clustering. We, on the other hand, focus on both this problem and the complementary question of when does such stability property even hold, either exactly or approximately. Related notions of clustering stability such as perturbation stability and approximation stability have also been studied. However, the goal in these works is to approximate a clustering objective such as k-means or k-median under stability assumptions on the optimizer of the objective (Ackerman and Ben-David, 2009; Awasthi et al., 2012; Bilu and Linial, 2012; Balcan et al., 2013; Balcan and Liang, 2016; Makarychev and Makarychev, 2016).

Closely related to our work is the paper by Daniely et al. (2012). They show that if there is an unknown ground-truth clustering with cluster size at least αn satisfying a slightly stronger requirement than IP-stability, i.e., each point, on average, is at least a factor 3 closer to the points in its own cluster than to the points in any other cluster, then they can find an $\mathcal{O}(1)$-approximately IP-stable clustering. However, their algorithm runs in time $n^{\Omega(\log 1/\alpha)} = \Omega(n^{\log k})$.

Individual Fairness and Fairness in Clustering In the fairness literature, fairness notions are commonly categorized into notions of individual fairness or group fairness. The latter ensures fairness for groups of individuals (such as men vs women), whereas the former aims to ensure fairness for *single* individuals. Our proposed notion of IP-stability can be interpreted as a notion of individual fairness for clustering. Individual fairness was originally proposed by Dwork et al. (2012) for classification, requiring that similar data points (as measured by a given task-specific metric) should receive a similar prediction. It has also been studied in the setting of online learning (Joseph et al., 2016; 2018; Gillen et al., 2018). Recently, two notions of individual fairness for clustering have been proposed. The first notion, proposed by Jung et al. (2020) and also studied by Mahabadi and Vakilian (2020), Chakrabarty and Negahbani (2021) and Vakilian and Yalciner (2022), asks that every data point is somewhat close to a center, where "somewhat" depends on how close the data point is to its $\lceil n/k \rceil$ nearest neighbors. Jung et al. motivate their notion from the perspective of facility location, but it can also be seen in the context of data points that strive to be well represented by being close to a center similarly to the notion of Chen et al. (2019) discussed above. Note that our proposed notion of IP-stability defines "being well represented" in terms of the average distance of a data point to the other points in its cluster, rather than the distance to a center, and hence is not restricted to centroid-based clustering. The second notion, proposed by Anderson et al. (2020), is analogous to the notion of individual fairness by Dwork et al. (2012) for classification. It considers probabilistic clustering, where each data point is mapped to a distribution over a set of centers, and requires that similar data points are mapped to similar distributions. Interestingly, Anderson et al. allow the similarity measure used to define fairness and the similarity measure used to evaluate clustering quality to be the same or different. However, individual fairness as introduced by Dwork et al. (2012) has often been deemed impractical due to requiring access to a task-specific metric (Ilvento, 2019), and it is unclear where a fairness similarity measure that is different from the clustering similarity measure should come from. Note that both notions are different from our notion: (1) there exists clusterings that are IP-stable, but will achieve $\alpha = \infty$ according α-fairness notion of Jung et al. and (2) the distributional property of Anderson et al. is satisfied with uniform distribution whereas our stability clusterings do not always exist.

Hedonic Games A related line of work is the class of hedonic games as a model of coalition formation (Dreze and Greenberg, 1980; Bogomolnaia and Jackson, 2002; Elkind et al., 2016). In a hedonic game the utility of a player only depends on the identity of players belonging to her coalition. Feldman et al. (2015b) study clustering from the perspective of hedonic games. Our work is different from theirs in the sense that in our model the data points are not selfish players.

3. NP-Hardness

We show that in general metrics, the problem of deciding whether an IP-stable k-clustering exists is NP-hard. For such a result it is crucial to specify how an input instance is encoded: we assume that a data set $\mathcal{D}$ together with a distance function d is represented by the distance matrix $(d(x, y))_{x,y \in \mathcal{D}}$. Under this assumption we can prove the following theorem:

Theorem 1 (NP-hardness of IP-stable clustering)**.** *Deciding whether a data set $\mathcal{D}$ together with a distance function d has an IP-stable k-clustering is NP-hard. This holds even if $k = 2$ and d is a metric distance.*

Due to space limitation, the proof of this theorem and all other missing proofs are deferred to the appendix.

The proof is via a reduction from a variant of 3-SAT where the number of clauses is equal to the number of variables and each variable occurs in at most three clauses. Unless P = NP, Theorem 1 implies that for general data sets, even when being guaranteed that an IP-stable k-clustering exists, there cannot be any efficient algorithm for computing such an IP-stable clustering. However, as with all NP-hard problems, there are two possible remedies. The first remedy is to look at approximately IP-stable clusterings:

Definition 2 (Approximate IP-stability)**.** *Let $\mathcal{C} = (C_1, \dots, C_k)$ be a k-clustering of $\mathcal{D}$ and for $x \in \mathcal{D}$ let $C(x)$ be the cluster C_i that x belongs to. We say that for some $t \geq 1$, $x \in \mathcal{D}$ is t-approximately IP-stable if either $C(x) = \{x\}$ or*

$$\frac{1}{|C(x)| - 1} \sum_{y \in C(x)} d(x, y) \leq \frac{t}{|C_i|} \sum_{y \in C_i} d(x, y) \qquad (2)$$

for every $C_i \neq C(x)$. The clustering $\mathcal{C}$ is t-approximately IP-stable if every $x \in \mathcal{D}$ is t-approximately IP-stable.

An alternative way of defining approximate IP-stability, which is explored in Section 4.1, would be to allow a violation of inequality (1) in Definition 1 for a certain number of points. In our experiments in Section 6 we will actually consider both these notions of approximation.

The second remedy is to restrict our considerations to data sets with some special structure. This is what we do in Section 5 where we consider the 1-dimensional Euclidean metric and tree metrics.

To complement Theorem 1, we show theoretical lower bounds for the standard clustering algorithms k-means++, k-center, and single linkage. Their violation of IP-stability can be arbitrarily large.

Theorem 2. *For any $\alpha > 1$, there exist separate examples where the clusterings produced by k-means++, k-center, and single linkage algorithms are t-approximately IP-stable only for $t \geq \alpha$.*

4. Approximation Algorithms for IP-Stability

In this section, we provide two algorithms for finding approximately IP-stable clustering. Our first algorithm finds a partially IP-stable clustering and the second one finds an approximately IP-stable clustering if the input point set admits a clustering that satisfies a set of requirements that are slightly stronger than the requirements of IP-stability.

4.1. Partially Stable Clustering

Here, we show that if we only want to cluster $(1-\varepsilon)$-fraction of the points, it is possible to find a $\mathcal{O}(\frac{\log^2 n}{\varepsilon})$-approximation for IP-stable k-clustering in any metric space.

We first define *hierarchically well-separated trees* (HSTs).

Definition 3 (Bartal (1996))**.** *A t-hierarchically well-separated tree (t-HST) is defined as a rooted weighted tree with the following properties:*

- *The edge weight from any node to each of its children is the same.*
- *The edge weight along any path from the root to a leaf are decreasing by a factor of at least t.*

Fakcharoenphol et al. (2004) shows that it is possible to have a tree embedding where distortion is $\mathcal{O}(\log n)$ in expectation. Recently, Haeupler et al. (2021) proposes an algorithm that give a worst-case distortion, which they call *partial tree embeddings* if it is allowed to remove a constant fraction of points from the embedding. While their work concerns hop-constrained network design, we provide a simplified version of their result which is useful in our context.

Theorem 3 (Haeupler et al. (2021))**.** *Given weighted graph $G = (V, E, w)$, $0 < \varepsilon < \frac{1}{3}$ and root $r \in V$, there is a polynomial-time algorithm which samples from a distribution over partial tree embeddings whose trees are 2-HST and rooted at r with exclusion probability*[2] *ε and worst-case distance stretch $\mathcal{O}(\frac{\log^2 n}{\varepsilon})$.*

Claim 4. *Without loss of generality, we can assume the following two properties from the construction in Theorem 3: (1) V is the set of leaves of T, and (2) $depth(u) = depth(v)$ for any $u, v \in V$.*

Our remaining task is to find an IP-stable clustering of *leaves* (according to the tree metric distance). If we can do that, then it follows that we have an $\mathcal{O}(\log^2(n)/\varepsilon)$-approximately IP-stable k-clustering.

Claim 5. *There exists a k-clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of leaves on t-HST for any $t \geq 2$ such that $\mathcal{C}$ is IP-stable for any leaf node. Moreover, for $u, v \in C_i$ and $w \in C_j$, $d_T(u, v) \leq d_T(u, w)$.*

By combining the tree embedding result and our clustering on HSTs we have the following theorem.

Theorem 6. *Let (V, d) be a metric. There is a randomized, polynomial time algorithm that produces a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ for $V' \subseteq V$, where V' is taken from V with exclusion probability ε such that, for any node $u \in C_i, j \neq i$, $\overline{d}(u, C_i) \leq \mathcal{O}(\frac{\log^2 n}{\varepsilon})\overline{d}(u, C_j)$* [3]*.*

Proof. We use Theorem 3 to embed (V, d) into a 2-HST (V', T). We then apply Claim 5 to produce $(C_1, \ldots, C_k)$ that is IP-stable in T. Since T is a partial tree embedding with worst-case distortion $\mathcal{O}(\frac{\log^2 n}{\varepsilon})$, it follows that $\overline{d}(u, C_i) \leq \overline{d}_T(u, C_i) \leq \overline{d}_T(u, C_j) \leq \mathcal{O}(\frac{\log^2 n}{\varepsilon})\overline{d}(u, C_j)$. $\square$

4.2. Instances with Stable Clustering

Next, we show an algorithm that finds an approximately IP-stable clustering, if there is a *well-separated* underlying clustering. Let us first define a well-separated clustering. A similar notion is also defined by Daniely et al. (2012).

Definition 4 ((α, γ)-clustering)**.** *Let $\mathcal{C} = (C_1, \ldots, C_k)$ be a k-clustering of $\mathcal{D}$, that is $\mathcal{D} = C_1 \dot\cup \ldots \dot\cup C_k$ and $C_i \neq \emptyset$ for $i \in [k]$. For $x \in \mathcal{D}$, we write $C(x)$ for the cluster C_i that x belongs to. We say that $\mathcal{C}$ is (α, γ)-clustering if*

1. *For all C_i, $|C_i| \geq \alpha \cdot n$, where $n = |\mathcal{D}|$, and*
2. *For all $i \neq j$ and $x \in C_i$, $\overline{d}(x, C_j) \geq \gamma \cdot \overline{d}(x, C_i)$.*

Lemma 1 (Daniely et al. (2012))**.** *Let $\mathcal{C} = (C_1, \ldots, C_k)$ be an (α, γ)-clustering, and let $i \neq j$. Then*

1. *For every*[4] *$x \in C_i, y \in C_j$, $\frac{\gamma-1}{\gamma}\overline{d}(y, C_i) \leq d(x, y) \leq \frac{\gamma^2+1}{\gamma(\gamma-1)}\overline{d}(y, C_i)$, and*
2. *For every $x, y \in C_i$, $d(x, y) \leq \frac{2}{\gamma-1}\overline{d}(x, C_j)$.*

Here, we show that, for large enough γ, the edges inside a cluster will always be smaller than edges between clusters.

Claim 7. *Let $\mathcal{C} = (C_1, \ldots, C_k)$ be an (α, γ)-clustering. Let $i \neq j$, and let $x, y \in C_i$ and $z \in C_j$. If $\gamma \geq 2 + \sqrt{3}$, then $d(x, y) \leq d(y, z)$.*

Theorem 8. *Let $\gamma \geq 2 + \sqrt{3}$. If there exists an (α, γ)-clustering, then there is an algorithm that finds such a clustering in time $\mathcal{O}(n^2 \log n + n \cdot \left(\frac{1}{\alpha}\right)^k)$.*

[2]Let $\mathcal{D}$ be distribution of partial tree metrics of G. $\mathcal{D}$ has *exclusion probability* ϵ if for all $v \in V$, we have $\Pr_{d\sim\mathcal{D}}[v \in V_d] \geq 1 - \epsilon$

[3]We let $\overline{d}(u, C) = \sum_{v\in C} \frac{d(u,v)}{|C\setminus\{u\}|}$ be the average distance from u to cluster C. If $|C| = \{u\}$, then $\overline{d}(u, C) = 0$.

[4]Daniely et al. (2012) use the term *almost every* to avoid sets of measurement zero.

Proof. Let us consider a modification to *single-linkage* algorithm: We consider edges in non-decreasing order and only merge two clusters if at least one of them has size at most αn. Claim 7 suggests that the edges within an underlying cluster will be considered before edges between two clusters. Since we know that any cluster size is at least αn, when considering an edge, it is safe to use this condition to merge them. If it is not the case, we can ignore the edge.

By this process, we will end up with a clustering where each cluster has size at least αn. Hence, there are at most $\mathcal{O}(1/\alpha)$ such clusters. We then can enumerate over all possible clusterings, as there are at most $\mathcal{O}(\frac{1}{\alpha^k})$ such clusterings, the running time follows. □

Note that the algorithm described in the above theorem has a high running time (especially if α is small). Notice that, before the enumeration, we do not make any mistakes when we run the modified single-linkage, and we get a clustering where each cluster has size at least αn. We further show it is possible to define a metric over this clustering and apply Theorem 6 to the metric. The clustering we get will be approximately IP-stable.

Conditioning the clusters via single-linkage When we run the single-linkage algorithm, in addition to the size of any pair of clusters, we can also consider the ratio between each pair of points in two different clusters. We want it so that distances for every pairs of points in two clusters are roughly the same. Moreover, the distance from a point x to its own cluster should not be large compared to the distance from x to any other clusters. We formally define this condition below.

Claim 9. *Let $\mathcal{C} = (C_1, \ldots, C_k)$ be an (α, γ)-clustering. Let $D \subseteq C_i$ and $D' \subseteq C_j$ be two set of points from different clusters. Then for $x \in D$ and $y, y' \in D'$, $\frac{d(x,y)}{d(x,y')} \leq \frac{\gamma^2+1}{(\gamma-1)^2}$.*

Corollary 1. *For two subsets D, D' from different underlying clusters, let $x, x' \in D$ and $y, y' \in D'$. Then*

$$\frac{d(x,y)}{d(x',y')} \leq \left(\frac{\gamma^2+1}{(\gamma-1)^2}\right)^2.$$

Claim 10. *Let D, D' be two clusters we consider in the single-linkage algorithm. Let $e = (x, y)$ be an edge that we merge in an arbitrary step of single-linkage algorithm where $x \in D, y \in D'$, then D and D' must belong to the same underlying clustering if one of the followings is true.*

1. *$|D| < \alpha \cdot n$ or $|D'| < \alpha \cdot n$,*
2. *$\frac{\max_{x\in D, y\in D'} d(x,y)}{\min_{x\in D, y\in D'} d(x,y)} > \left(\frac{\gamma^2+1}{(\gamma-1)^2}\right)^2$,*
3. *There exists $x' \in D$ such that $d(x, x') > \frac{2\gamma}{(\gamma-1)^2} d(x, y)$.*

Theorem 11. *Let $\alpha > 0, \gamma \geq 2 + \sqrt{3}$. If there exists an (α, γ)-clustering, then there is a randomized algorithm that finds a $\mathcal{O}(\frac{\log^2(1/\alpha)}{\alpha})$-approximately IP-stable k-clustering in polynomial time and constant success probability.*

Proof. In the first phase, we run the modified single-linkage algorithm: As in the standard single-linkage algorithm, we consider edges in a non-decreasing order of length and merge two clusters using the criteria of Claim 10. When this phase finishes, we get a clustering $(D_1, \ldots, D_\ell)$ with $k \leq \ell \leq \mathcal{O}(\frac{1}{\alpha})$, where (1) $|D_i| \geq \alpha n$, (2) For any $x \in D_i, y \in D_j$, $d(x,y)$ approximates $\overline{d}(D_i, D_j)$[5] , and (3) $\overline{d}(x, D_i) = \mathcal{O}(\overline{d}(x, D_j))$.

This implies that we can pick $\mathcal{R} = \{r_i, \ldots, r_\ell\}$ where each $r_i \in D_i$ is the representative of D_i. Then, we show an IP-stable clustering of $\mathcal{R}$ yields an IP-stable clustering of the initial point set as well.

We apply Theorem 3 to $\mathcal{R}$ with exclusion probability $\varepsilon < \alpha$ to produce a partial tree embedding T. By the choice of ε, with constant probability, not a single point in $\mathcal{R}$ is excluded We then apply Claim 5 to produce a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of $\mathcal{R}$. Let $\mathcal{F} = (F_1, \ldots, F_k)$ where $F_i = \bigcup_{r_j \in C_i} D_j$ be the final clustering. Next, we show that $\mathcal{F}$ is an approximately IP-stable clustering.

For every $x \in F_i$, let y be the point furthest to x in F_i and z be the point closest to x in F_j. If we show that $d(x,y) = \mathcal{O}(\frac{\log^2(1/\alpha)}{\alpha}) d(x,z)$, then this proves the claim as $\overline{d}(x, F_i) = \mathcal{O}(d(x,y))$ and $\overline{d}(x, F_j) = \Omega(d(x,z))$.

Let D_x, D_y, D_z be the clusters x, y, z belong to after the merging phases terminates, respectively. Also, let r_x, r_y, r_z be the representatives of D_x, D_y, D_z. By Claim 5, since r_x, r_y belong to the same cluster C, and since r_z belongs to another cluster C', $d(r_x, r_y) \leq d_T(r_x, r_y) \leq d_T(r_x, r_z) \leq \mathcal{O}(\frac{\log^2(1/\alpha)}{\alpha}) d(r_x, r_z)$ where the last inequality holds since T is a 2-HST embedding for the representative points with worst-case distortion guarantee of $\mathcal{O}(\frac{\log^2(1/\alpha)}{\alpha})$. Because of Claim 10, $d(x,y) = \Theta(d(r_x, r_y))$ and $d(x,z) = \Theta(d(r_x, r_z))$, it follows that $d(x,y) = \mathcal{O}(\frac{\log^2(1/\alpha)}{\alpha}) d(x,z)$. Hence, the proof is complete. □

Remark Our analysis relies on the fact that the partial embedding provides a worst-case guarantee. However, to the best of our knowledge, a probabilistic tree embedding does not seem to work because the distortion guarantee is on expectation. In other words, there could be an edge e such that the distortion is bad, and having only one such edge is enough to break the stability of any k-clustering on the tree embedding.

[5] $\overline{d}(C, C') = \sum_{x\in C, y\in C'} \frac{d(x,y)}{|C||C'|}$.

5. Special Metrics

Another approach to circumvent the NP-hardness of IP-stable clustering is to restrict our considerations to data sets with some special structure. Here, we consider 1-dimensional Euclidean metrics and tree metrics.

5.1. 1-dimensional Euclidean Case

Here we study the special case of $\mathcal{D} \subseteq \mathbb{R}$ where $|\mathcal{D}| = n$, and d is the Euclidean metric. We provide an $\mathcal{O}(kn)$ algorithm that finds an IP-stable k-clustering. We show the following Theorem 12 holds.

High-level Idea We start with a specific clustering where all but one cluster are singletons. By the definition, only nodes in the non-singleton cluster may want to *leave* the cluster (i.e., the IP-stability condition may be violated for a subset of points in that cluster). Lemma 5 shows that if a node v of a cluster C desires to leave C, then a boundary node v' of C will want to leave C as well. This allows us to focus only on boundary vertices. There can be at most two vertices per cluster, as long as our clustering is contiguous. We further observe that a cluster will become unstable only when an additional vertex joins the cluster. When this happens, since the vertex that just joined, which has become a boundary vertex, does not desire to leave the cluster, it must be the boundary vertex on the other end that wants to leave the cluster. Since we maintain the order and the monotonicity of clusters and boundary vertices that we inspect, we end up with $\mathcal{O}(kn)$ time algorithm. We show that the following theorem holds.

Theorem 12. *Let $\mathcal{D} \subseteq \mathbb{R}$ be a point set of size n. There exists an algorithm that for any $1 \leq k \leq n$, gives an IP-stable k-clustering of $\mathcal{D}$, and has a time complexity of $\mathcal{O}(kn)$.*

It is well-known that for any set of n points in a metric space, there exists an embedding to one-dimensional Euclidean space with distortion $\mathcal{O}(n)$. Hence:

Corollary 2. *Let $\mathcal{D}$ be a point set of size n in an arbitrary metric space. There exists a polynomial time algorithm that returns an $\mathcal{O}(n)$-approximately IP-stable k-clustering of $\mathcal{D}$.*

In Appendix F, we consider an extension of the 1-dimensional Euclidean case where a target size for each cluster is given, and the goal is to find an IP-stable k-clustering that minimizes the total violation from the target cluster sizes. Formally, the goal is to solve

$$\min_{\substack{\mathcal{C}=(C_1,\ldots,C_k)\colon \mathcal{C} \text{ is an} \\ \text{IP-stable clustering of } \mathcal{D} \\ \text{with contiguous clusters}}} \|(|C_1| - t_1, \ldots, |C_k| - t_k)\|_p, \quad (3)$$

where $t_1, \ldots, t_k \in [n]$ with $\sum_{i=1}^{k} t_i = n$ are given target cluster sizes, $p \in \mathbb{R}_{\geq 1} \cup \{\infty\}$ and $\|\cdot\|_p$ denotes the ℓ_p-norm. In Appendix F, we provide an $\mathcal{O}(n^3 k)$ dynamic programming approach for this problem.

5.2. IP-Stable Clusterings on Trees

In this section, we show how to find an IP-stable 2-clustering when d is a tree metric. Let $T = (V, E)$ be a given weighted tree rooted at r (r can be arbitrarily chosen). Our construction will first pick a *boundary edge* among edges adjacent to r. We then *rotate* the boundary edge in a systematic manner until the clustering we have, which are two connected components we get by removing the boundary edge e from T, is IP-stable. While it could be the case that non-contiguous IP-stable clustering exists, we only consider contiguous clustering in our algorithm. Our algorithm implies that such a clustering exists.

For any graph $G = (V, E)$ and $v \in V$, let $C_G(v)$ be the set of vertices reachable from v in G. For any tree $T = (V, E)$ and an edge $e \in E$, let $T \setminus e = (V, E \setminus \{e\})$ be the subgraph of T obtained by removing e. Let $N(u)$ denote the set of neighbors of u in T. We say that $u^f \in N(u)$ is the *furthest neighbor* of u if $\overline{d}(u, C_{T\setminus(u,u^f)}(u^f)) \geq \overline{d}(u, C_{T\setminus(u,w)}(w))$ for any $w \in N(u)$. Next, we define a rotate operation which is crucial for our algorithm and show that the following property holds:

Claim 13. *Suppose the current boundary edge is (u, v), we define the operation rotate(u) to be an operation that moves the boundary edge from (u, v) to (u, u^f). After rotate(u) is called, the clustering defined by the boundary edge (u, u^f) is stable for u.*

Algorithm Now we are ready to describe our algorithm. First, pick an edge $(b_0 = r, v)$ arbitrarily among edges adjacent to the root r, and call rotate(b_0). After rotation, let $e_1 = (b_0, b_1)$ denote the boundary edge. By Claim 13, the clustering defined by e_1 is stable for b_0. If it is also stable for b_1, then we have a stable clustering. If not, suppose the boundary edge is now $e_i = (b_{i-1}, b_i)$. By induction, we assume that the clustering is stable for b_{i-1}. If this clustering is not stable, then it is not stable for b_i. We call rotate(b_i) to move the boundary edge to $e_{i+1} = (b_i, b_{i+1})$. b_{i+1} cannot be b_{i-1}, otherwise, the previous clustering would already be stable for b_i. We can repeat this process until we find a stable clustering. The process will terminate because the boundary edge is moved further away from the root at every step. Eventually, we will reach the point where $b_i^f = b_{i-1}$. This might happen at a leaf. Once it happens, then we have a stable clustering for the boundary nodes. In Appendix D.3, we show that when considering a contiguous clustering, if the boundary nodes are stable, then every node is stable. In other words:

Claim 14. *Let $e = (u, v)$ be the boundary edge of a 2-clustering of T. If u and v are stable, then the clustering is* *stable for every node $x \in V$.*

Finally, since the boundary edge is moving away from a fixed vertex, we call rotate at most T times.

6. Experiments

We ran a number of experiments.[6] Our experiments are intended to serve as a proof of concept. They do not focus on the running times of the algorithms or their applicability to *large* data sets. Hence, we only use rather small data sets of sizes 500 to 1885.

Let us define some quantities. For any point x, let

$$\mathrm{Vi}(x) = \max_{C_i \neq C(x)} \frac{\frac{1}{|C(x)|-1}\sum_{y\in C(x)} d(x,y)}{\frac{1}{|C_i|}\sum_{y \in C_i} d(x,y)},$$

where we use the convention that $\frac{0}{0} = 0$. A point x is IP-stable if and only if $\mathrm{Vi}(x) \leq 1$. Let $U = \{x \in \mathcal{D} : x \text{ is not stable}\}$. We measure the extent to which a k-clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of a dataset $\mathcal{D}$ is (un-)stable by $\#\mathrm{Uns} = |U|$ ("number of unstable"), $\mathrm{MaxVi} = \max_{x\in\mathcal{D}} \mathrm{Vi}(x)$ ("maximum violation"), and MeanVi ("mean violation") defined as

$$\mathrm{MeanVi} = \begin{cases} \frac{1}{|U|}\sum_{x\in U} \mathrm{Vi}(x) & U \neq \emptyset \\ 0 & U = \emptyset \end{cases}.$$

The clustering $\mathcal{C}$ is IP-stable if and only if $\#\mathrm{Uns} = 0$ and $\mathrm{MaxVi} \leq 1$. MaxVi is the smallest value of t such that $\mathcal{C}$ is a t-approximately IP-stable clustering. We measure the quality of $\mathcal{C}$ w.r.t. the goal of putting similar points into the same cluster by Co ("cost"), defined as the average within-cluster distance. Formally,

$$\mathrm{Co} = \sum_{i=1}^{k} \frac{1}{\binom{|C_i|}{2}} \sum_{\{x,y\}\in C_i \times C_i} d(x,y). \tag{4}$$

We performed all experiments in Python. We used the standard clustering algorithms from Scikit-learn[7] or SciPy[8] with all parameters set to their default values.

6.1. General Data Sets

We performed the same set of experiments on the first 1000 records of the Adult data set, the Drug Consumption data set (1885 records), and the Indian Liver Patient data set (579 records), which are all publicly available in the UCI repository (Dua and Graff, 2019). As distance function d we used the Euclidean, Manhattan or Chebyshev metric. Here we only present the results for the Adult data set on the Euclidean metric, the other results are provided in Appendix G. Our observations are largely consistent between the different data sets and metrics.

[6]Code available on https://github.com/amazon-research/ip-stability-for-clustering
[7]https://scikit-learn.org/
[8]https://scipy.org/

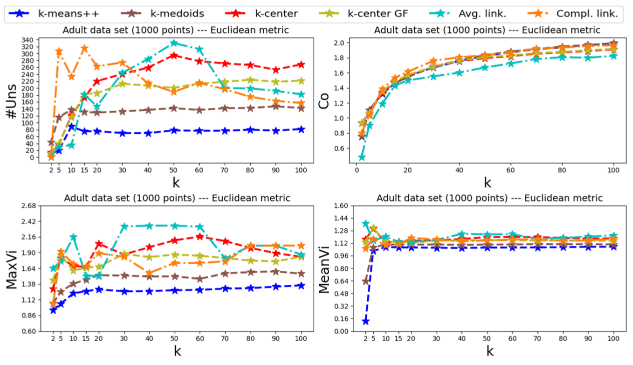

Figure 3. # Uns **(top-left)**, Co **(top-right)**, MaxVi **(bottom-left)** and MeanVi **(bottom-right)** for the clusterings produced by the various algorithms as a function of k. k-center GF denotes the group-fair k-center algorithm of Kleindessner et al. (2019a).

(Un-)Stability of Standard Algorithms Working with the Adult data set, we only used its six numerical features (e.g., age, hours worked per week), normalized to zero mean and unit variance, for representing records. We applied several standard clustering algorithms as well as the group-fair k-center algorithm of Kleindessner et al. (2019a) (referred to as k-center GF) to the data set (k-means++; k-medoids) or its distance matrix (k-center using the greedy strategy of Gonzalez (1985); k-center GF; single / average / complete linkage clustering). In order to study the extent to which these methods produce (un-)stable clusterings, for $k = 2, 5, 10, 15, 20, 30, \ldots, 100$, we computed # Uns, MaxVi and MeanVi as defined above for the resulting k-clusterings. For measuring the quality of the clusterings we computed Co as defined in (4).

Figure 3 shows the results, where we removed the single linkage algorithm since it performs significantly worse than the other algorithms. For k-means++, k-medoids, k-center and k-center GF we show average results obtained from running them for 25 times since their outcomes depend on random initializations. We can see that, in particular for large values of k, k-center, k-center GF, and the linkage algorithms can be quite unstable with rather large values of # Uns and MaxVi. In contrast, k-means++ produces quite stable clusterings with $\#\mathrm{Uns} \leq 90$ and $\mathrm{MaxVi} \leq 1.4$ even when k is large. For a baseline comparison, for a random clustering in which every data point was assigned to one of $k = 100$ clusters uniformly at random we observed $\#\mathrm{Uns} = 990$, $\mathrm{MaxVi} = 11.0$, and $\mathrm{MeanVi} = 2.5$ on average. The k-medoids algorithm performs worse than

k-means++, but better than the other algorithms. The clusterings produced by k-center GF, which we ran with the constraint of choosing $\lfloor k/2 \rfloor$ female and $\lceil k/2 \rceil$ male centers, are slightly more stable than the ones produced by k-center. However, note that it really depends on the data set whether group-fairness correlates with IP-stability or not (see Appendix G.1). All methods perform similarly w.r.t. the clustering cost Co.

Heuristics to Improve Standard Algorithms One might wonder whether we can modify the standard clustering algorithms in order to make them more stable. A natural idea to make any clustering more stable is to change it locally and to iteratively pick a data point that is not stable and assign it to the cluster that it is closest to. However, this idea turned out not to work as we observed that usually we can only pick a very small number of data points whose reassignment does not cause other data points that are initially stable to become unstable after the reassignment (see Appendix G.2). Another idea that we explored is specifically tied to linkage clustering. We provide the details in Appendix G.3.

6.2. 1-dimensional Euclidean Data Sets

In Appendix H we present experiments with our DP approach on real world 1-dimensional Euclidean data sets.

7. Discussion

We proposed a notion of IP-stability that aims at data points being well represented by their clusters. Formally, it requires every point, on average, to be closer to the points in its own cluster than to the points in any other cluster. IP-stability is not restricted to centroid-based clustering and raises numerous questions, some of which we addressed: in a general metric space, we showed it is NP-hard to decide whether an IP-stable k-clustering exists and we provided approximation algorithms for the problem when the input contains a "well-separated" clustering or when a small fraction of inputs can be excluded in the output. For one-dimensional Euclidean data sets we can compute an IP-stable clustering in polynomial time. For the case of tree metrics, we showed how to find an IP-stable 2-clustering in polynomial time. We proved that in the worst-case scenario some standard clustering algorithms including k-means++ provide arbitrarily unstable clusterings. However, our experiments show that k-means++ works quite well in practice.

While our works focus mostly on the upper bound side, understanding the lower bound, e.g., hardness of approximation for IP-stability is one important open question. It is a natural direction to explore whether our ideas for IP-stable 2-clustering on trees can be extended to k-clustering for $k > 2$? Finally, it would be very interesting to provide theoretical evidence supporting our empirical findings that show the surprising effectiveness of k-means++ for our proposed notion of IP-stability, in spite of the existence of bad worst-case scenarios.

Acknowledgements

SA is supported by the Simons Collaborative grant on Theory of Algorithmic Fairness, and the National Science Foundation grant CCF-1733556. JM is supported by fundings from the NSF AI Institute for the Foundations of Machine Learning (IFML), an NSF Career award, and the Simons Collaborative grant on Theory of Algorithmic Fairness. AV is supported by NSF award CCF-1934843.

References

M. Abbasi, A. Bhaskara, and S. Venkatasubramanian. Fair clustering via equitable group representations. In *ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT)*, 2021.

M. Ackerman and S. Ben-David. Clusterability: A theoretical study. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

S. Ahmadi, S. Galhotra, B. Saha, and R. Schwartz. Fair correlation clustering. arXiv:2002.03508 [cs.DS], 2020.

S. Ahmadian, A. Epasto, R. Kumar, and M. Mahdian. Clustering without over-representation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2019.

S. Ahmadian, A. Epasto, M. Knittel, R. Kumar, M. Mahdian, B. Moseley, P. Pham, S. Vassilvitskii, and Y. Wang. Fair hierarchical clustering. In *Neural Information Processing Systems (NeurIPS)*, 2020a.

S. Ahmadian, A. Epasto, R. Kumar, and M. Mahdian. Fair correlation clustering. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020b.

A. Anagnostopoulos, L. Becchetti, M. Böhm, A. Fazzone, S. Leonardi, C. Menghini, and C. Schwiegelshohn. Principal fairness: Removing bias via projections. arXiv:1905.13651 [cs.DS], 2019.

N. Anderson, S. Bera, S. Das, and Y. Liu. Distributional individual fairness in clustering. arXiv:2006.12589 [cs.LG], 2020.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Symposium on Discrete Algorithms (SODA)*, 2007.

P. Awasthi and M.-F. Balcan. Center based clustering: A foundational perspective. In *Handbook of Cluster Analysis*. CRC Press, 2014.

P. Awasthi, A. Blum, and O. Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1-2):49–54, 2012.

A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner. Scalable fair clustering. In *International Conference on Machine Learning (ICML)*, 2019.

M.-F. Balcan and Y. Liang. Clustering under perturbation resilience. *SIAM Journal on Computing*, 45(1):102–155, 2016.

M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *ACM Symposium on Theory of Computing (STOC)*, 2008.

M.-F. Balcan, A. Blum, and A. Gupta. Clustering under approximation stability. *Journal of the ACM (JACM)*, 60 (2):1–34, 2013.

M.-F. Balcan, T. Dick, R. Noothigattu, and A. Procaccia. Envy-free classification. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Symposium on Foundations of Computer Science (FOCS)*, 1996.

S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani. Fair algorithms for clustering. In *Neural Information Processing Systems (NeurIPS)*, 2019.

I. O. Bercea, M. Groß, S. Khuller, A. Kumar, C. Rösner, D. R. Schmidt, and M. Schmidt. On the cost of essentially fair clusterings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, 2019.

Y. Bilu and N. Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(5):643–660, 2012.

A. Bogomolnaia and M. O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38 (2):201–230, 2002.

B. Brubach, D. Chakrabarti, J. Dickerson, S. Khuller, A. Srinivasan, and L. Tsepenekas. A pairwise fair and community-preserving approach to k-center clustering. In *International Conference on Machine Learning (ICML)*, 2020.

B. Brubach, D. Chakrabarti, J. Dickerson, A. Srinivasan, and L. Tsepenekas. Fairness, semi-supervised learning, and more: A general framework for clustering with stochastic pairwise constraints. In *AAAI Conference on Artificial Intelligence*, 2021.

M. E. Celebi and K. Aydin. *Unsupervised Learning Algorithms*. Springer, 2016.

D. Chakrabarty and M. Negahbani. Better algorithms for individually fair k-clustering. *arXiv preprint arXiv:2106.12150*, 2021.

X. Chen, B. Fain, L. Lyu, and K. Munagala. Proportionally fair clustering. In *International Conference on Machine Learning (ICML)*, 2019.

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

E. Chlamtáč, Y. Makarychev, and A. Vakilian. Approximating fair clustering with cascaded norm objectives. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2022.

A. Daniely, N. Linial, and M. Saks. Clustering is difficult only when it does not matter. arXiv:1205.4891 [cs.LG]], 2012.

S. Dasgupta. Performance guarantees for hierarchical clustering. In *International Conference on Computational Learning Theory (COLT)*, 2002.

I. Davidson and S. S. Ravi. Making existing clusterings fairer: Algorithms, complexity results and insights. In *AAAI Conference on Artificial Intelligence*, 2020.

J. H. Dreze and J. Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica: Journal of the Econometric Society*, pages 987–1003, 1980.

D. Dua and C. Graff. UCI machine learning repository, 2019. German Credit data set available on https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data). Adult data set available on https://archive.ics.uci.edu/ml/datasets/adult. Drug Consumption data set available on https://archive.ics.uci.edu/ml/datasets/Drug+consumption+(quantified). Indian Liver Patient data set available on https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset).

C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Innovations in Theoretical Computer Science Conference (ITCS)*, 2012.

E. Elkind, A. Fanelli, and M. Flammini. Price of pareto optimality in hedonic games. In *AAAI Conference on Artificial Intelligence*, 2016.

S. Esmaeili, B. Brubach, L. Tsepenekas, and J. Dickerson. Probabilistic fair clustering. In *Neural Information Processing Systems (NeurIPS)*, 2020.

J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

E. Fehrman, A. K. Muhammad, E. M. Mirkes, V. Egan, and A. N. Gorban. The five factor model of personality and evaluation of drug consumption risk. arXiv:1506.06297 [stat.AP], 2015. Data set available on `https://archive.ics.uci.edu/ml/datasets/Drug+consumption+(quantified)`.

M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015a.

M. Feldman, L. Lewin-Eytan, and J. Naor. Hedonic clustering games. *ACM Trans. Parallel Comput.*, 2(1), 2015b.

D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3): 223–232, 1985.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

M. Ghadiri, S. Samadi, and S. Vempala. Socially fair k-means clustering. In *ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT)*, 2021.

S. Gillen, C. Jung, M. Kearns, and A. Roth. Online learning with an unknown fairness metric. In *Neural Information Processing Systems (NeurIPS)*, 2018.

T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38: 293–306, 1985.

B. Haeupler, E. Hershkowitz, and G. Zuzic. Tree embeddings for hop-constrained network design. In *Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2021.

E. Harb and H. Lam. KFC: a scalable approximation algorithm for k-center fair clustering. In *Neural Information Processing Systems (NeurIPS)*, 2020.

L. Huang, S. H.-C. Jiang, and N. K. Vishnoi. Coresets for clustering with fairness constraints. In *Neural Information Processing Systems (NeurIPS)*, 2019.

C. Ilvento. Metric learning for individual fairness. arXiv:1906.00250 [cs.LG], 2019.

M. Joseph, M. Kearns, J. Morgenstern, and A. Roth. Fairness in learning: Classic and contextual bandits. In *Neural Information Processing Systems (NIPS)*, 2016.

M. Joseph, M. Kearns, J. Morgenstern, S. Neel, and A. Roth. Meritocratic fairness for infinite and contextual bandits. In *AAAI / ACM Conference on Artificial Intelligence, Ethics, and Society*, 2018.

C. Jung, S. Kannan, and N. Lutz. A center in your neighborhood: Fairness in facility location. In *Symposium on Foundations of Responsible Computing (FORC)*, 2020.

M. Kleindessner, P. Awasthi, and J. Morgenstern. Fair k-center clustering for data summarization. In *International Conference on Machine Learning (ICML)*, 2019a.

M. Kleindessner, S. Samadi, P. Awasthi, and J. Morgenstern. Guarantees for spectral clustering with fairness constraints. In *International Conference on Machine Learning (ICML)*, 2019b.

S. Lloyd. Least squares quantization in pcm. *EEE Transactions on Information Theory*, 28(2):129–137, 1982.

S. Mahabadi and A. Vakilian. Individual fairness for k-clustering. In *International Conference on Machine Learning (ICML)*, 2020.

K. Makarychev and Y. Makarychev. Metric perturbation resilience. *arXiv preprint arXiv:1607.06442*, 2016.

Y. Makarychev and A. Vakilian. Approximation algorithms for socially fair clustering. In *Conference on Learning Theory (COLT)*, 2021.

E. Micha and N. Shah. Proportionally fair clustering revisited. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, 2020.

R. Ostrovsky, Y. Rabani, L. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59(6):1–22, 2013.

C. Rösner and M. Schmidt. Privacy preserving clustering with constraints. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2018.

A. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of political Economy*, 92(6):991–1016, 1984.

M. Schmidt, C. Schwiegelshohn, and C. Sohler. Fair coresets and streaming algorithms for fair k-means clustering. arXiv:1812.10854 [cs.DS], 2018.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

A. Vakilian and M. Yalciner. Improved approximation algorithms for individually fair clustering. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning (ICML)*, 2001.

Appendix

A. Other Related Work

Other notions of fairness for clustering Fair clustering was first studied in the seminal work of Chierichetti et al. (2017). It is based on the fairness notion of disparate impact (Feldman et al., 2015a), which says that the output of a machine learning algorithm should be independent of a sensitive attribute, and asks that each cluster has proportional representation from different demographic groups. Chierichetti et al. provide approximation algorithms that incorporate their notion into k-median and k-center clustering, assuming that there are only two demographic groups. Several follow-up works extend this line of work to other clustering objectives such as k-means or spectral clustering, multiple or non-disjoint groups, some variations of the fairness notion, to address scalability issues or to improve approximation guarantees (Rösner and Schmidt, 2018; Schmidt et al., 2018; Ahmadian et al., 2019; Anagnostopoulos et al., 2019; Backurs et al., 2019; Bera et al., 2019; Bercea et al., 2019; Huang et al., 2019; Kleindessner et al., 2019b; Ahmadian et al., 2020a;b; Ahmadi et al., 2020; Esmaeili et al., 2020; Harb and Lam, 2020). The recent work of Davidson and Ravi (2020) shows that for two groups, when given any clustering, one can efficiently compute the fair clustering (fair according to the notion of Chierichetti et al.) that is most similar to the given clustering using linear programming. Davidson and Ravi also show that it is NP-hard to decide whether a data set allows for a fair clustering that additionally satisfies some given must-link constraints. They mention that such must-link constraints could be used for encoding individual level fairness constraints of the form "similar data points must go to the same cluster". However, for such a notion of individual fairness it remains unclear which pairs of data points exactly should be subject to a must-link constraint.

Alternative fairness notions for clustering are tied to centroid-based clustering such as k-means, k-median and k-center, where one chooses k centers and then forms clusters by assigning every data point to its closest center: (i) Motivated by the application of data summarization, Kleindessner et al. (2019a) propose that the various demographic groups should be proportionally represented among the chosen centers. (ii) Chen et al. (2019) propose a notion of proportionality that requires that no sufficiently large subset of data points could jointly reduce their distances from their closest centers by choosing a new center. This notion is similar to our notion of IP-stability in that it assumes that an individual data point strives to be well represented (in the notion of Chen et al. by being close to a center) and it also does not rely on demographic group information. However, while our notion aims at ensuring stability for every single data point, the notion of Chen et al. only looks at sufficiently large subsets. Micha and Shah (2020) further studied the proportionally fair clustering in Euclidean space and on graph metrics. (iii) Brubach et al. (2020; 2021) introduce the notions of pairwise fairness and community preserving fairness and incorporate them into k-center clustering. (iv) Ghadiri et al. (2021) and Abbasi et al. (2021) study a fair version of k-means clustering where the goal is to minimize the maximum average cost that a demographic group incurs (the maximum is over the demographic groups and the average is over the various data points within a group). Makarychev and Vakilian (2021) and Chlamtáč et al. (2022) designed optimal algorithms for minimizing the global clustering cost of a clustering (e.g., k-median and k-means cost) with respect to this notion of fairness and its generalization known as clustering with cascaded norms.

B. Proof of Theorem 1

We show NP-hardness of the IP-stable clustering decision problem (with $k = 2$ and d required to be a metric) via a reduction from a variant of 3-SAT. It is well known that deciding whether a Boolean formula in conjunctive normal form, where each clause comprises at most three literals, is satisfiable is NP-hard. NP-hardness also holds for a restricted version of 3-SAT, where each variable occurs in at most three clauses (Garey and Johnson, 1979, page 259). Furthermore, we can require the formula to have the same number of clauses as number of variables as the following transformation shows: let Φ be a formula with m clauses and n variables. If $n > m$, we introduce $l = \lfloor \frac{n-m+1}{2} \rfloor$ new variables $x_1, \ldots, x_l$ and for each of them add three clauses (x_i) to Φ (if $n - m$ is odd, we add only two clauses (x_l)). The resulting formula has the same number of clauses as number of variables and is satisfiable if and only if Φ is satisfiable. Similarly, if $n < m$, we introduce $l = \lfloor 3 \cdot \frac{m-n}{2} + \frac{1}{2} \rfloor$ new variables $x_1, \ldots, x_l$ and add to Φ the clauses $(x_1 \vee x_2 \vee x_3), (x_4 \vee x_5 \vee x_6), \ldots, (x_{l-2} \vee x_{l-1} \vee x_l)$ (if $m - n$ is odd, the last clause is $(x_{l-1} \vee x_l)$ instead of $(x_{l-2} \vee x_{l-1} \vee x_l)$). As before, the resulting formula has the same number of clauses as number of variables and is satisfiable if and only if Φ is satisfiable.

So let $\Phi = C_1 \wedge C_2 \wedge \ldots \wedge C_n$ be a formula in conjunctive normal form over variables $x_1, \ldots, x_n$ such that each clause C_i comprises at most three literals x_j or $\neg x_j$ and each variable occurs in at most three clauses (as either x_j or $\neg x_j$). We construct a metric space $(\mathcal{D}, d)$ in time polynomial in n such that $\mathcal{D}$ has an IP-stable 2-clustering with respect to d if and only if Φ is satisfiable (for n sufficiently large). We set

$$\mathcal{D} = \{True, False, \star, \infty, C_1, \ldots, C_n, x_1, \neg x_1, \ldots, x_n, \neg x_n\}$$

and

$$d(x,y) = [d'(x,y) + \mathbb{1}\{x \neq y\}] + \mathbb{1}\{x \neq y\} \cdot \max_{x,y \in \mathcal{D}} [d'(x,y)+1], \quad x,y \in \mathcal{D},$$

for some symmetric function $d' : \mathcal{D} \times \mathcal{D} \to \mathbb{R}_{\geq 0}$ with $d'(x,x) = 0$, $x \in \mathcal{D}$, that we specify in the next paragraph. It is straightforward to see that d is a metric. Importantly, note that for any $x \in \mathcal{D}$, inequality (1) holds with respect to d if and only if it holds with respect to d'.

We set $d'(x,y) = 0$ for all $x, y \in \mathcal{D}$ except for the following:

$$\begin{aligned}
d'(True, False) &= A, \\
d'(True, \star) &= B, \\
d'(\star, False) &= C, \\
d'(C_i, False) &= D, \quad i = 1, \ldots, n, \\
d'(C_i, \star) &= E, \quad i = 1, \ldots, n, \\
d'(\infty, True) &= F, \\
d'(\infty, False) &= G, \\
d'(\infty, \star) &= H, \\
d'(C_i, \infty) &= J, \quad i = 1, \ldots, n, \\
d'(x_i, \neg x_i) &= S, \quad i = 1, \ldots, n, \\
d'(C_i, \neg x_j) &= U, \quad (i,j) \in \{(i,j) \in \{1, \ldots, n\}^2 : x_j \text{ appears in } C_i\}, \\
d'(C_i, x_j) &= U, \quad (i,j) \in \{(i,j) \in \{1, \ldots, n\}^2 : \neg x_j \text{ appears in } C_i\},
\end{aligned}$$

where we set

$$\begin{aligned}
&A = n, \qquad F = n^2, \qquad B = 2F = 2n^2, \qquad E = \frac{5}{2}F = \frac{5}{2}n^2, \\
&J = E + \log n = \frac{5}{2}n^2 + \log n, \qquad D = J + \log^2 n = \frac{5}{2}n^2 + \log n + \log^2 n, \\
&U = 3J = \frac{15}{2}n^2 + 3\log n, \qquad H = nD + E = \frac{5}{2}n^3 + \frac{5}{2}n^2 + n\log n + n\log^2 n, \\
&G = H + 2n^2 - n - 2n\log^2 n = \frac{5}{2}n^3 + \frac{9}{2}n^2 - n + n\log n - n\log^2 n, \\
&S = (3n+3)U = \frac{45}{2}n^3 + \frac{45}{2}n^2 + 9n\log n + 9\log n, \\
&C = \frac{A + G + nD}{2} = \frac{5}{2}n^3 + \frac{9}{4}n^2 + n\log n.
\end{aligned} \tag{5}$$

We show that for $n \geq 160$ there is a satisfying assignment for Φ if and only if there is an IP-stable 2-clustering of $\mathcal{D}$.

- *"Satisfying assignment $\Rightarrow$ IP-stable 2-clustering"*

  Let us assume we are given a satisfying assignment of Φ. We may assume that if x_i only appears as x_i in Φ and not as $\neg x_i$, then x_i is true; similarly, if x_i only appears as $\neg x_i$, then x_i is false. We construct a clustering of $\mathcal{D}$ into two clusters V_1 and V_2 as follows:

$$\begin{aligned}
V_1 &= \{True, \infty, C_1, \ldots, C_n\} \cup \{x_i : x_i \text{ is true in sat. ass.}\} \cup \{\neg x_i : \neg x_i \text{ is true in sat. ass.}\}, \\
V_2 &= \{False, \star\} \cup \{x_i : x_i \text{ is false in satisfying assignment}\} \cup \{\neg x_i : \neg x_i \text{ is false in sat. ass.}\}.
\end{aligned}$$

  It is $|V_1| = 2 + 2n$ and $|V_2| = 2 + n$. We need show that every data point in $\mathcal{D}$ is stable.This is equivalent to verifying that the following inequalities are true:

Points in V_1:

$$True: \quad \frac{1}{1+2n}\sum_{v\in V_1} d'(True,v) = \frac{F}{1+2n} \leq \frac{A+B}{2+n} = \frac{1}{2+n}\sum_{v\in V_2} d'(True,v) \tag{6}$$

$$\infty: \quad \frac{1}{1+2n}\sum_{v\in V_1} d'(\infty,v) = \frac{F+nJ}{1+2n} \leq \frac{G+H}{2+n} = \frac{1}{2+n}\sum_{v\in V_2} d'(\infty,v) \tag{7}$$

$$C_i: \quad \frac{1}{1+2n}\sum_{v\in V_1} d'(C_i,v) \leq \frac{J+2U}{1+2n} \leq \frac{U+D+E}{2+n} \leq \frac{1}{2+n}\sum_{v\in V_2} d'(C_i,v) \tag{8}$$

$$x_i: \quad \frac{1}{1+2n}\sum_{v\in V_1} d'(x_i,v) \leq \frac{2U}{1+2n} \leq \frac{S}{2+n} \leq \frac{1}{2+n}\sum_{v\in V_2} d'(x_i,v) \tag{9}$$

$$\neg x_i: \quad \frac{1}{1+2n}\sum_{v\in V_1} d'(\neg x_i,v) \leq \frac{2U}{1+2n} \leq \frac{S}{2+n} \leq \frac{1}{2+n}\sum_{v\in V_2} d'(\neg x_i,v) \tag{10}$$

Points in V_2:

$$False: \quad \frac{1}{1+n}\sum_{v\in V_2} d'(False,v) = \frac{C}{1+n} \leq \frac{A+G+nD}{2+2n} = \frac{1}{2+2n}\sum_{v\in V_1} d'(False,v) \tag{11}$$

$$\star: \quad \frac{1}{1+n}\sum_{v\in V_2} d'(\star,v) = \frac{C}{1+n} \leq \frac{B+H+nE}{2+2n} = \frac{1}{2+2n}\sum_{v\in V_1} d'(\star,v) \tag{12}$$

$$x_i: \quad \frac{1}{1+n}\sum_{v\in V_2} d'(x_i,v) = 0 \leq \frac{S}{2+2n} \leq \frac{1}{2+2n}\sum_{v\in V_1} d'(x_i,v) \tag{13}$$

$$\neg x_i: \quad \frac{1}{1+n}\sum_{v\in V_2} d'(\neg x_i,v) = 0 \leq \frac{S}{2+2n} \leq \frac{1}{2+2n}\sum_{v\in V_1} d'(\neg x_i,v) \tag{14}$$

It is straightforward to check that for our choice of A,B,C,D,E,F,G,H,J,S,U as specified in (5) all inequalities (6) to (14) are true.

- *"IP-stable 2-clustering $\Rightarrow$ satisfying assignment"*

  Let us assume that there is an IP-stable clustering of $\mathcal{D}$ with two clusters V_1 and V_2. For any partitioning of $\{C_1,\dots,C_n\}$ into two sets of size l and $n-l$ ($0\leq l\leq n$) we denote the two sets by $\mathcal{C}_l$ and $\widetilde{\mathcal{C}}_{n-l}$.

  We first show that x_i and $\neg x_i$ cannot be contained in the same cluster (say in V_1). This is because if we assume that $x_i, \neg x_i \in V_1$, for our choice of S and U in (5) we have

$$\frac{1}{|V_2|}\sum_{v\in V_2} d'(x_i,v) \leq U < \frac{S}{3n+2} \leq \frac{1}{|V_1|-1}\sum_{v\in V_1} d'(x_i,v)$$

  in contradiction to x_i being stable. As a consequence we have $n\leq |V_1|, |V_2| \leq 2n+4$.

  Next, we show that due to our choice of A,B,C,D,E,F,G,H,J in (5) none of the following cases can be true:

  1. $\{True,\infty\}\cup\mathcal{C}_l \subset V_1$ and $\widetilde{\mathcal{C}}_{n-l}\cup\{False,\star\}\subset V_2$ for any $0\leq l<n$

     In this case, $False$ would not be stable since for all $0\leq l<n$,

$$\frac{1}{|V_1|}\sum_{v\in V_1} d'(False,v) = \frac{A+G+lD}{l+2+n} < \frac{C+(n-l)D}{n-l+1+n} = \frac{1}{|V_2|-1}\sum_{v\in V_2} d'(False,v).$$

  2. $\{True\}\cup\mathcal{C}_l \subset V_1$ and $\widetilde{\mathcal{C}}_{n-l}\cup\{False,\star,\infty\}\subset V_2$ for any $0\leq l\leq n$

In this case, $False$ would not be stable since for all $0 \leq l \leq n$,

$$\frac{1}{|V_1|} \sum_{v \in V_1} d'(False, v) = \frac{A + lD}{l + 1 + n} < \frac{C + G + (n-l)D}{n - l + 2 + n} = \frac{1}{|V_2| - 1} \sum_{v \in V_2} d'(False, v).$$

3. $\{False, \infty\} \cup \mathcal{C}_l \subset V_1$ and $\widetilde{\mathcal{C}}_{n-l} \cup \{True, \star\} \subset V_2$ for any $0 \leq l \leq n$

   In this case, $True$ would not be stable since for all $0 \leq l \leq n$,

$$\frac{1}{|V_1|} \sum_{v \in V_1} d'(True, v) = \frac{A + F}{l + 2 + n} < \frac{B}{n - l + 1 + n} = \frac{1}{|V_2| - 1} \sum_{v \in V_2} d'(True, v).$$

4. $\{False\} \cup \mathcal{C}_l \subset V_1$ and $\widetilde{\mathcal{C}}_{n-l} \cup \{True, \star, \infty\} \subset V_2$ for any $0 \leq l \leq n$

   In this case, $True$ would not be stable since for all $0 \leq l \leq n$,

$$\frac{1}{|V_1|} \sum_{v \in V_1} d'(True, v) = \frac{A}{l + 1 + n} < \frac{B + F}{n - l + 2 + n} = \frac{1}{|V_2| - 1} \sum_{v \in V_2} d'(True, v).$$

5. $\{\star, \infty\} \cup \mathcal{C}_l \subset V_1$ and $\widetilde{\mathcal{C}}_{n-l} \cup \{False, True\} \subset V_2$ for any $0 \leq l \leq n$

   In this case, $\star$ would not be stable since for all $0 \leq l \leq n$,

$$\frac{1}{|V_2|} \sum_{v \in V_2} d'(\star, v) = \frac{B + C + (n-l)E}{n - l + 2 + n} < \frac{H + lE}{l + 1 + n} = \frac{1}{|V_1| - 1} \sum_{v \in V_1} d'(\star, v).$$

6. $\{\star\} \cup \mathcal{C}_l \subset V_1$ and $\widetilde{\mathcal{C}}_{n-l} \cup \{False, True, \infty\} \subset V_2$ for any $0 \leq l \leq n$

   In this case, ∞ would not be stable since for all $0 \leq l \leq n$,

$$\frac{1}{|V_1|} \sum_{v \in V_1} d'(\infty, v) = \frac{H + lJ}{l + 1 + n} < \frac{F + G + (n-l)J}{n - l + 2 + n} = \frac{1}{|V_2| - 1} \sum_{v \in V_2} d'(\infty, v).$$

7. $\mathcal{C}_l \subseteq V_1$ and $\widetilde{\mathcal{C}}_{n-l} \cup \{True, False, \star, \infty\} \subseteq V_2$ for any $0 \leq l \leq n$

   In this case, $True$ would not be stable since for all $0 \leq l \leq n$,

$$\frac{1}{|V_1|} \sum_{v \in V_1} d'(True, v) = 0 < \frac{A + B + F}{3 + (n-l) + n} = \frac{1}{|V_2| - 1} \sum_{v \in V_2} d'(True, v).$$

8. $\{\infty\} \cup \mathcal{C}_l \subseteq V_1$ and $\widetilde{\mathcal{C}}_{n-l} \cup \{True, False, \star\} \subseteq V_2$ for any $0 \leq l \leq n$

   In this case, $True$ would not be stable since for all $0 \leq l \leq n$,

$$\frac{1}{|V_1|} \sum_{v \in V_1} d'(True, v) = \frac{F}{1 + l + n} < \frac{A + B}{2 + (n-l) + n} = \frac{1}{|V_2| - 1} \sum_{v \in V_2} d'(True, v).$$

Of course, in all these cases we can exchange the role of V_1 and V_2. Hence, $True, \infty, C_1, \ldots, C_n$ must be contained in one cluster and $\star, False$ must be contained in the other cluster. W.l.o.g., let us assume $True, \infty, C_1, \ldots, C_n \in V_1$ and $\star, False \in V_2$ and hence $|V_1| = 2n + 2$ and $|V_2| = n + 2$.

Finally, we show that for the clause $C_i = (l_j)$ or $C_i = (l_j \vee l_{j'})$ or $C_i = (l_j \vee l_{j'} \vee l_{j''})$, with the literal l_j equaling x_j or $\neg x_j$, it cannot be the case that $C_i, \neg l_j$ or $C_i, \neg l_j, \neg l_{j'}$ or $C_i, \neg l_j, \neg l_{j'}, \neg l_{j''}$ are all contained in V_1. This is because otherwise

$$\frac{1}{|V_2|}\sum_{v\in V_2} d'(C_i,v) = \frac{D+E}{n+2} < \frac{U+J}{2n+1} \leq \frac{1}{|V_1|-1}\sum_{v\in V_1} d'(C_i,v) \tag{15}$$

for our choice of D, E, J, U in (5) and C_i would not be stable. Consequently, since x_j and $\neg x_j$ are not in the same cluster, for each clause C_i at least one of its literals must be in V_1.

Hence, if we set every literal x_i or $\neg x_i$ that is contained in V_1 to a true logical value and every literal x_i or $\neg x_i$ that is contained in V_2 to a false logical value, we obtain a valid assignment that makes Φ true. □

C. Missing Proofs from Section 4

C.1. Proof of Claim 4

Proof. The idea is to extend nodes corresponding to actual points in V so that they are the leaves and that they are at the same depth. The distortion of the modified tree will be worse by a constant factor, but that is fine for our purpose as the construction has $\mathcal{O}(polylog(n))$ distortion. Let (V', d_T) be a given partial tree metric from the construction in Theorem 6 and let $\ell = \text{depth}(d_T)$ be the depth of the tree. If a node $v \in V$ is not a leaf of T, we augment T by replacing v with v' and connecting v' to v with a path $v', v_1, v_2, \ldots, v_{\ell-\text{depth}_T(v)-1}, v$. Let T' be the tree after our modification. By this augmentation, v is now a leaf in T' and $\text{depth}_{T'}(v) = \ell - 1$. Let $(V'', d_{T'})$ be the new tree metric. For any $u \in V$, we show that $d_{T'}(u,v) \leq 3d_T(u,v)$. In words, this is because the weight of the added path between v' and v is a telescopic sum, and this sums up to at most the weight of the edge connecting v to its parent in T, i.e. $p_T(v)$. That is, $d_{T'}(u,v) = d_{T'}(u,v') + d_{T'}(v',v) = d_T(u,v) + d_{T'}(v',v) \leq d_T(u,v) + d_{T'}(p_{T'}(v'),v') = d_T(u,v) + d_T(p_T(v),v) \leq 3d_T(u,v)$. Since we can apply this argument to any node, the claim holds true. □

C.2. Proof of Claim 5

Proof. Let T be our HST tree rooted at r. Let $S = \{v_1, v_2 \ldots, v_k\}$ be a set of nodes that we are going to pick. Suppose they are sorted in such a way that for every $1 \leq i < k$, $\text{depth}(v_i) \geq \text{depth}(v_{i+1})$. We let $\text{depth}(r) = 0$, and for every node u, $\text{depth}(u) = \text{depth}(p(u)) + 1$ where $p(u)$ is the parent of u in T.

To find S, consider the lowest depth, i.e., furthest from the root, ℓ such that the number of nodes at depth ℓ is at most k. Let V_ℓ be the set of nodes at depth ℓ. Initially, $S = \emptyset$. For each $v \in V_\ell$, if $|S| + |\text{children}(v)| + |V_\ell| - 1 < k$, then add $\text{children}(v)$ to S and remove v from V_ℓ and continue. Otherwise, if $|S| + |\text{children}(v)| + |V_\ell| - 1 = k$, then add $\text{children}(v)$ and $V_\ell - \{v\}$ to S. Since $|S| = k$ after this operation, we are done. In the last case where $|S| + |\text{children}(v)| + |V_\ell| - 1 > k$, select a subset of $\text{children}(v)$ of size $k - |S| + |V_\ell|$. Let this subset be V_v. Let $S = S \cup V_\ell \cup V_v$. An example is given in Figure 4 for $k = 4$.

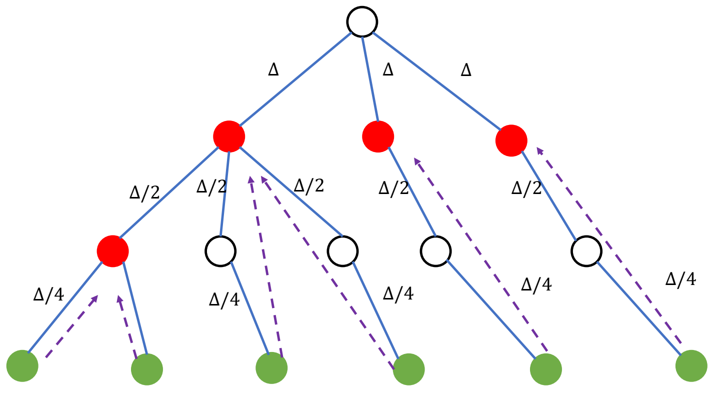

Figure 4. IP-stable 4-clustering on a 2-HST. S is the set of the red nodes.

For $i \leq k$, let T_i to be the subtree of $T \setminus \left(\bigcup_{j<i} T_j\right)$ rooted at v_i. Let C_i denote the i^{th} cluster consisting of the leaves of T_i. We argue that $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ is an IP-stable clustering of T. Let u, v be two leaves in T. Let x

be the lowest common ancestor (lca) of u and v. Since T is an HST tree, and since all leaves are at the same depth, $d_T(u,v) = d_T(u,x) + d_T(x,v) = 2d(u,x)$. For any pair of $C_i \in \mathcal{C}, u \in C_i$ where u is a leaf, as we show above, the distance from u to any other leaf node $v \in C_i$ is at most $2d_T(u,v_i)$. This is because the lca of u and v can either be v_i or its descendant. Hence, we can say that $\frac{1}{|C_i|-1}\sum_{w\in C_i} d_T(u,w) \leq 2d_T(u,v_i)$. Moreover, for any leaf node $v' \in C_{j\neq i}$, we have that $d_T(u,v') \geq 2d_T(u,v_i)$. This is because the lca of u and v' can either be v_i or its ancestor. Hence, $\frac{1}{|C_j|}\sum_{w\in C_j} d_T(u,w) \geq 2d_T(u,v_i)$. Therefore, $\frac{1}{|C_i|-1}\sum_{w\in C_i} d(u,w) \leq \frac{1}{|C_j|}\sum_{w\in C_j} d_T(u,w)$, and $\mathcal{C}$ is a stable clustering for T.

We end the proof by noting that for any $u, v \in C_i$ and $w \in C_j$, $d_T(u,v) \leq 2d_T(u,v_i) \leq d_T(u,v_j) \leq d_T(u,w)$. □

C.3. Proof of Claim 7

Proof.

$$\begin{aligned} d(x,y) &\leq \frac{2}{\gamma-1}\bar{d}(y,C_j) && \big(\text{By (2) in Lemma 1}\big) \\ &\leq \frac{\gamma-1}{\gamma}\bar{d}(y,C_j) && \big(\gamma \geq 2+\sqrt{3}\big) \\ &\leq d(y,z) && \big(\text{By (1) in Lemma 1}\big) \end{aligned}$$

□

C.4. Proof of Theorem 8

Proof. Let us consider a modification to *single-linkage* algorithm: We consider edges in non-decreasing order and only merge two clusters if one (or both) size is consisting of at most αn points. Claim 7 suggests that the edges within an underlying cluster will be considered before edges between two clusters. Since we know that any cluster size is at least αn, when considering an edge, it is safe to use this condition to merge them. If it is not the case, we can ignore the edge.

By this process, we will end up with a clustering where each cluster has size at least αn. There are at most $\mathcal{O}(1/\alpha)$ such clusters. We then can enumerate over all possible clusterings, as there are at most $\mathcal{O}(\frac{1}{\alpha^k})$ such clusterings, the running time follows. □

C.5. Proof of Claim 9

Proof. From Lemma 1, $\frac{\gamma-1}{\gamma}\bar{d}(x,C_j) \leq d(x,y), d(x,y') \leq \frac{\gamma^2+1}{\gamma(\gamma-1)}\bar{d}(x,C_j)$. Hence,

$$\frac{d(x,y)}{d(x,y')} \leq \frac{\gamma^2+1}{\gamma(\gamma-1)} \cdot \frac{\gamma}{\gamma-1} = \frac{\gamma^2+1}{(\gamma-1)^2}.$$

□

C.6. Proof of Claim 10

Proof. (1) follows from Claim 7 and the fact that we consider edges in non-decreasing order, hence, when we consider e, if D, D' belong to different underlying clusters, and if $|D| < \alpha n$, then an edge e' within the cluster C_D must already be considered. At that point, D should be merged into another cluster, hence a contradiction.

(2) follows from Corollary 1. If $\frac{\max_{x\in D, y\in D'} d(x,y)}{\min_{x\in D, y\in D'} d(x,y)} > \left(\frac{\gamma^2+1}{(\gamma-1)^2}\right)^2$, then there exists $x, x' \in D$ and $y, y' \in D'$ that violate the inequality in Corollary 1, so D and D' must belong to the same underlying cluster.

It remains to show (3). Suppose D belongs to the underlying cluster C_D and D' belongs to another underlying cluster $C_{D'}$, then $d(x,y) \geq \frac{\gamma-1}{\gamma}\bar{d}(x,C_{D'})$ ((1) in Lemma 1). However,

$$\begin{aligned} d(x,x') &> \frac{2\gamma}{(\gamma-1)^2}d(x,y) > \frac{2\gamma}{(\gamma-1)^2} \cdot \frac{\gamma-1}{\gamma}\bar{d}(x,C_{D'}) \\ &= \frac{2}{\gamma-1}\bar{d}(x,C_{D'}). \end{aligned}$$

Since we know that x, x' belong to the same underlying cluster, this is a contradiction. $\square$

D. Missing Proofs of Section 5

D.1. Proof of Theorem 12

Proof. Initially, start with the leftmost cluster containing $n - k + 1$ points, and all remaining $k - 1$ clusters having one single point. Imagine clusters are separated with some separators. An example is given in Figure 5 for $k = 4$. We show that by only moving the separators to the left, an IP-stable k-clustering can be found. In this proof, we mainly argue about the

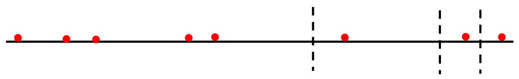

Figure 5. Initialization for $k = 4$ and $n = 8$.

stability of boundary nodes of clusters; however, Lemma 5 in Appendix F shows that stability of boundary nodes implies stability of all the nodes. Since all the clusters but the first one are singletons, and singleton clusters are IP-stable, the only separator that might want to move is the first separator. The first separator moves to the left until the first cluster becomes IP-stable. In the next step, the only cluster that is not IP-stable is the second cluster since the first cluster has just been made IP-stable, and the rest are singletons. The second cluster is unstable due to some nodes on its right hand side that want to join the third cluster by moving the second separator to the left (note that the leftmost node of the second cluster is stable since the first separator has just moved). If the second cluster shrinks, some rightmost nodes of the first cluster might get unstable, and want to join the second cluster by moving the first separator to the left. The same scenario repeats over and over again until the first two clusters get stable by moving the first two separators only to the left. Using the same argument, at each step i, the only cluster that is not IP-stable is the i^{th} cluster, and the first i clusters can be made stable by moving the first i separators only to the left.

Consider the last, i.e. k^{th} step. If the k^{th} cluster is a singleton, then it is IP-stable, and our algorithm has converged to an IP-stable k-clustering. Assume the k^{th} cluster is not a singleton. The reason that it is not a singleton is that the points on its left hand side had preferred to move from the $(k-1)^{th}$ cluster to the k^{th} cluster. If the k^{th} cluster is a better cluster for its leftmost point, so is a better cluster for all other points inside it. Therefore, the k^{th} cluster is an IP-stable cluster, and our algorithm converges to an IP-stable k-clustering.

By using the fact that if a separator moves it only moves to the left, and therefore, each separator moves at most n times, in Lemma 2 in Appendix D.2, we show the running time of the algorithm is $\mathcal{O}(kn)$. $\square$

D.2. Running Time of the Algorithm for 1-dimensional Euclidean Metric

Lemma 2. *The running time of the algorithm described for the* 1*-d Euclidean metric case is* $\mathcal{O}(kn)$.

Proof. First, we argue that if the following information are maintained for each cluster, the stability of each boundary node can be checked in $\mathcal{O}(1)$.

- sum of distances from the left-most and right-most nodes to all other nodes in the cluster
- the number of nodes in the cluster

Suppose that u is the right-most node of a cluster C_i. In order to check if u is stable, we compare the average distance from u to its own cluster (C_i) with the average distance from u to the adjacent cluster, say (C_{i+1}). Let v be the node adjacent to u in C_{i+1}, then $\overline{d}(u, C_{i+1}) = d(u, v) + \overline{d}(v, C_{i+1})$. Since $d(u, C_i), d(v, C_{i+1}), |C_i|, |C_{i+1}|$ are all maintained, we can check if $\overline{d}(u, C_i) \leq \overline{d}(u, C_{i+1})$ in $\mathcal{O}(1)$. The argument for the case that u is the leftmost node of a cluster C_i holds in a similar way. At the beginning of the algorithm, since our initialization has only one big cluster, coming up with the needed information for two boundary nodes of the big cluster takes $\mathcal{O}(n)$. Since the only operation that our algorithm does is to move a separator to the left, we show whenever such an operation happens, all the information stored that need to get updated, can get updated in $\mathcal{O}(1)$. Suppose when a separator moves to the left, the right-most node of C_i, let's say u, moves to C_{i+1}. Let s, t, v, w be the left-most node of C_i, the node next to u in C_i, the node next to u in C_{i+1}, and the right-most node of C_{i+1} before the move, respectively. Also, let $\tilde{C}_i = C_i \setminus \{u\}$ and $\tilde{C}_{i+1} = C_{i+1} \cup \{u\}$ denote the updated clusters C_i and C_{i+1} after u joins the cluster on its right. As $|\tilde{C}_i| = |C_i| - 1$ and $|\tilde{C}_{i+1}| = |C_{i+1}| + 1$, updating the numbers of nodes in two clusters is trivial. Now we show how to update other information, namely, $d(s, \tilde{C}_i), d(t, \tilde{C}_i), d(u, \tilde{C}_{i+1}), d(w, \tilde{C}_{i+1})$.

$$\begin{aligned} d(s, \tilde{C}_i) &= d(s, C_i) - d(s, u) \\ d(t, \tilde{C}_i) &= d(u, C_i) - |\tilde{C}_i| d(t, u) \\ d(u, \tilde{C}_{i+1}) &= d(u, C_{i+1}) = d(v, C_{i+1}) + |\tilde{C}_{i+1}| d(u, v) \\ d(w, \tilde{C}_{i+1}) &= d(w, C_{i+1}) + d(u, w) \end{aligned}$$

As the right-hand sides of the relationships above are maintained each update can be performed in $\mathcal{O}(1)$. Since in the algorithm each separator only moves to the left, the total number of times that separators are moved is at most kn. Each time that a separator wants to move, the stability of the node on its right is checked in $\mathcal{O}(1)$. Also, if a separator moves, updating the information stored takes $\mathcal{O}(1)$. Hence, the total time complexity of the algorithm is $\mathcal{O}(kn)$. □

D.3. Proof of Claim 14

Proof. Let C_1, C_2 be the clusters containing u and v, respectively. Wlog, suppose $|C_1| > 1$ and $x \in C_1$. Our goal is to show that $\overline{d}(x, C_1) \le \overline{d}(x, C_2)$.

Since the path from x to any $y \in C_2$ has to go through u, we have that $\overline{d}(x, C_2) = d(x, u) + \overline{d}(u, C_2)$.

As u is stable, $\overline{d}(u, C_1) \le \overline{d}(u, C_2)$.

If we can show that $\overline{d}(x, C_1) \le d(x, u) + \overline{d}(u, C_1)$, then we are done. This is true as

$$\begin{aligned} \overline{d}(x, C_1) &= \frac{1}{|C_1| - 1} \sum_{y \in C_1, y \neq x} d(x, y) \\ &= \frac{1}{|C_1| - 1} \sum_{y \in C_1, y \neq x} d(x, y) \\ &\le \frac{1}{|C_1| - 1} \sum_{y \in C_1, y \neq x} (d(x, u) + d(u, y)) \\ &= d(x, u) + \frac{1}{|C_1| - 1} \sum_{y \in C_1, y \neq x} d(u, y) \\ &\le d(x, u) + \frac{1}{|C_1| - 1} \sum_{y \in C_1} d(u, y) \\ &= d(x, u) + \overline{d}(u, C_1). \end{aligned}$$

□

E. Hard Instances for k-means++, k-center, and single linkage

In this section, we prove Theorem 2 by showing hard instances for k-means++, k-center, and single linkage clustering algorithms.

E.1. Hard Instances for k-means++

Here, we show that for any given approximation factor $\alpha > 1$, there exists an instance $\mathcal{I}_\alpha$ and a target number of clusters k_α such that with constant probability k-means++ outputs a k_α-clustering of $\mathcal{I}_\alpha$ that violates the requirement of IP-stable clustering by a factor of α.

High-level description of k-means++. k-means++ is an algorithm for choosing the initial seeds for the Lloyd's heuristic that provides a provable approximation guarantee (Arthur and Vassilvitskii, 2007). The high-level intuition is to spread out the initial set of k centers. The first center is picked uniformly at random, after which each subsequent center is picked from the remaining set points according to the probability distribution proportional to the squared distance of the points to the so-far-selected set of centers. Once the seeding phase picks k centers, k-means++ performs Lloyd's heuristic.

Before stating the argument formally, we show how to construct the hard instance $\mathcal{I}_\alpha$ for any given approximation parameter $\alpha > 1$ for the IP-stability requirement.

Structure of the hard instance. Given the approximation parameter α, $\mathcal{I}_\alpha$ is constructed as follows. The instance consists of n_α copies of block I such that each adjacent blocks are at distance D_α from each other; see Figure 6. As we explain in more detail, the solution SOL returned by k-means++ violates the IP-stability by a factor of α if there exists a block $I_j \in \mathcal{I}_\alpha$ such that $\text{SOL} \cap I_j = \{v_j, u_j\}$. In the rest of this section, our goal is to show that with constant probability this event happens when $k_\alpha = \frac{13}{12} n_\alpha$.

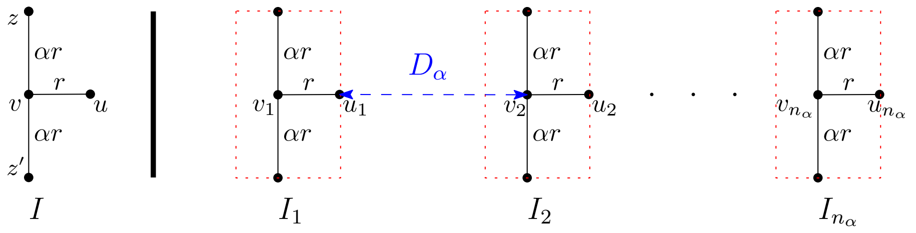

Figure 6. The construction of hard instance for a given parameter α. (a) shows the structure of a building block I and (b) shows a hard instance which constitutes of n_α blocks $I_1, \ldots, I_{n_\alpha}$.

Claim 15. *Suppose that v and u are picked as the initial set of centers for 2-clustering of I. Then, the solution returned by k-means++ violates the IP-stability requirement by a factor of α.*

Proof. It is straightforward to verify that once v and u are picked as the initial set of centers, after a round of Lloyd's heuristic v and u remain the cluster centers and k-means++ stops. This is the case since when v and u are centers initially the clusters are $\{z, z', v\}$ and $\{u\}$. The centroids of these two clusters are respectively v and u. Hence, k-means++ outputs $\{z, z', v\}$ and $\{u\}$ as the 2-clustering of I. The maximum violation of the IP-stable requirement for this 2-clustering corresponds to v which is equal to $\frac{(d(v,z)+d(v,z'))/2}{d(v,u)} = \alpha$. □

In this section, the squared distance function is denoted by Δ. Here is the main theorem of this section.

Theorem 16. *For any given parameter α, there exists an instance $\mathcal{I}_\alpha$ such that with constant probability the solution returned by k-means++ on $(\mathcal{I}_\alpha, k_\alpha = \frac{13}{12} n_\alpha)$ is violating the requirement of IP-stable clustering by a factor of α—the maximum violation over the points in $\mathcal{I}_\alpha$ is at least α.*

To prove the above theorem, we show that by setting the values of D_α and n_α properly, k-means++ with constant probability picks the points corresponding to u, v from at least one of the copies I_j (and picks no other points from I_j).

Lemma 3. *Let $\mathcal{S}_1$ denote the set of centers picked by k-means++ on $(\mathcal{I}_\alpha, \frac{13}{12} n_\alpha)$ after the first n_α iterations in the seeding phase where $n_\alpha \geq 18700$. If $D_\alpha > \alpha r \sqrt{\frac{3 n_\alpha}{\varepsilon}}$ where $\varepsilon < \frac{1}{100 n_\alpha}$, then with probability at least 0.98,*

1. *$\mathcal{S}_1$ contains exactly one point from each block; $\forall I \in \mathcal{I}_\alpha$, $|\mathcal{S}_1 \cap I| = 1$, and*
2. *In at least $\frac{n_\alpha}{10}$ blocks, copies of v are picked; $|j \in [n_\alpha] \mid I_j \cap \mathcal{S}_1 = \{v_j\}| \geq \frac{n_\alpha}{10}$.*

Proof. We start with the first property. Consider iteration $i \leq n_\alpha$ of k-means++ and let S_{i-1} denote the set of points that are selected as centers in the first $i-1$ iterations. Let $\mathcal{I}^+_{i-1} := \{I_j \mid I_j \cap S_{i-1} \neq \emptyset\}$ and $\mathcal{I}^-_{i-1} := \{I_j \mid I_j \cap S_{i-1} = \emptyset\}$. Since for any pair of points $p \in I, p' \in I'$ where $I \neq I'$, $\Delta(p, p') \geq D^2_\alpha$, for any point $p \in \mathcal{I}^-_{i-1}$, $\Delta(p, S_{i-1}) \geq D^2_\alpha$. Moreover, by the construction of the building block (see Figure 6-(a)), for any point $p' \in \mathcal{I}_{i-1}^{+}$, $\Delta(p', S_{i-1}) \leq 4\alpha^2 r^2 \leq \frac{4\varepsilon}{3n_\alpha} D_\alpha^2$. Hence, the probability that in iteration i the algorithm picks a point from $\mathcal{I}_{i-1}^{-}$ is

$$\frac{\sum_{p\in\mathcal{I}_{i-1}^{-}} \Delta(p, S_{i-1})}{\sum_{p\in\mathcal{I}_{i-1}^{-}} \Delta(p, S_{i-1}) + \sum_{p\in\mathcal{I}_{i-1}^{+}} \Delta(p, S_{i-1})} \geq \frac{4D_\alpha^2}{4D_\alpha^2 + 3n_\alpha \cdot (4\alpha^2 r^2)} \geq \frac{1}{1+\varepsilon} \geq 1-\varepsilon$$

Thus, the probability that i-th point is picked from $\mathcal{I}_{i-1}^{+}$ is at most ε. By union bound over the first n_α iterations of k-means++, the probability that $\mathcal{S}_1$ picks exactly one point from each block in $\mathcal{I}_\alpha$ is at least

$$1 - \sum_{i=1}^{n_\alpha} \Pr[i\text{-th point is picked from } \mathcal{I}_{i-1}^{+}] \geq 1 - \varepsilon \cdot n_\alpha \geq 0.99 \tag{16}$$

Next, we show that the second property also holds with high probability. Consider iteration i of the algorithm. By the approximate triangle inequality for Δ function[9] and since the distance of any pair of points within any block is $2\alpha r$, for a point $p \in I_j \in \mathcal{I}_{i-1}^{-}$, $\eta^2 \leq \Delta(p, S_{i-1}) \leq 2(\eta^2 + 4\alpha^2 r^2)$ where $\eta \geq D_\alpha$. In particular, for each block $I_j \in \mathcal{I}_{i-1}^{-}$,

$$\frac{\Delta(v_j, S_{i-1})}{\sum_{p\in I_j} \Delta(p, S_{i-1})} \geq \frac{\eta^2}{7\eta^2 + 24\alpha^2 r^2} \geq 1/(8 + \frac{\varepsilon}{n_\alpha}) \tag{17}$$

Let X_i be a random variable which indicates that the i-th point belongs to $\{v_j | j \in [n_\alpha]\}$; $X_i = 1$ if the i-th point belongs to $\{v_j | j \in [n_\alpha]\}$ and is equal to zero otherwise.

$$\begin{aligned}
\Pr[X_i = 1] &\geq \frac{\sum_{I_j\in\mathcal{I}_{i-1}^{-}} \Delta(v_j, S_{i-1})}{\sum_{I_j\in\mathcal{I}_{i-1}^{-}} \sum_{p\in I_j} \Delta(p, S_{i-1}) + \sum_{p\in\mathcal{I}_{i-1}^{+}} \Delta(p, S_{i-1})} \\
&\geq \frac{D_\alpha^2}{(8+\varepsilon/n_\alpha)\cdot D_\alpha^2 + \sum_{p\in\mathcal{I}_{i-1}^{+}} \Delta(p, S_{i-1})} && \triangleright \text{ by Eq. (17)} \\
&\geq \frac{D_\alpha^2}{(8+\varepsilon/n_\alpha)\cdot D_\alpha^2 + 3n_\alpha \cdot (4\alpha^2 r^2)} && \triangleright\ \forall p \in \mathcal{I}_{i-1}^{+}, \Delta(p, S_{i-1}) \leq 4\alpha^2 r^2 \\
&\geq \frac{D_\alpha^2}{(8+\varepsilon/n_\alpha)\cdot D_\alpha^2 + \varepsilon \cdot D_\alpha^2} > 1/9
\end{aligned}$$

Note that we showed that $\Pr[X_i = 1] \geq 1/9$ independent of the algorithm's choices in the first $i-1$ iterations of the algorithm. In particular, random variables $X_1, \ldots, X_{n_\alpha}$ are stochastically dominated by *independent* random variable $Y_1, \ldots, Y_{n_\alpha}$ where each $Y_i = 1$ with probability $1/9$ and is zero otherwise. Hence, the random variable $X := X_1 + \ldots + X_{n_\alpha}$ is stochasitcally dominated by Y which is $\mathrm{B}(n_\alpha, 1/9)$, the binomial distribution with n_α trials and success probability $1/9$. Hence, by an application of Hoeffding's inequality on Y,

$$\Pr[X \leq \frac{n_\alpha}{10}] \leq \Pr[Y \leq \frac{n_\alpha}{10}] \leq \exp(-2n_\alpha(\frac{1}{9} - \frac{1}{10})^2) \leq 0.01 \qquad \triangleright \text{ for } n_\alpha > 18700 \tag{18}$$

Thus, by Eq. (16) and (18), with probability at least 0.98 both properties hold. □

Lemma 4. *Let $\mathcal{S}$ be the set of centers picked at the end of the seeding phase of k-means++ on $(\mathcal{I}_\alpha, k_\alpha = \frac{13}{12} n_\alpha)$ where $n_\alpha > \max(9000, 240\alpha^2)$. Then, with probability at least 0.33, there exists a block $I_j \in \mathcal{I}_\alpha$ such that $\mathcal{S} \cap I_j = \{v_j, u_j\}$.*

Proof. Let $\mathcal{T}_1$ denote the event that $\mathcal{S}_1$, the set of centers picked in the first n_α iterations, picks exactly one point from each block and in at least $n_\alpha/10$ blocks the v-type points are picked in $\mathcal{S}_1$. Note that by an application of Lemma 3, $\Pr[\mathcal{T}_1] \geq 0.98$—with probability at least 0.98, $\mathcal{T}_1$ holds.

First we show the following. Let $m_\alpha = n_\alpha/20$. With constant probability, there exists an iteration $i^* \in [n_\alpha + 1, n_\alpha + m_\alpha]$ in which k-means++ picks a u-type point form I_j such that $I_j \cap S_{i^*-1} = \{v_j\}$. Let $\mathcal{E}_i$ denote the event that in iteration $i + n_\alpha$ where $i \in [m_\alpha]$, k-means++ picks the point u_j from I_j such that $S_{i-1} \cap I_j = \{v_j\}$. Note that for each block in

[9] $\forall u, v, w \in P, \Delta(v, w) \leq 2(\Delta(v, u) + \Delta(u, w))$.

$\{I_j \mid \mathcal{S}_1 \cap I_j = \{v_j\}\}$, $\sum_{p \in I_j} \Delta(p, \mathcal{S}_1) = (2\alpha^2+1)r^2$; otherwise, $\sum_{p\in I_j} \Delta(p, \mathcal{S}_1) \leq (5\alpha^2+1)r^2$. Hence, for each $i \in [m_\alpha]$, conditioned on $\mathcal{T}_1$ and independent of the prior choices of the algorithm in iterations $n_\alpha+1, \dots, i^* := n_\alpha + i - 1$,

$$\begin{aligned}
\Pr[\mathcal{E}_i | \mathcal{T}_1] &= \frac{\sum_{\{I_j \mid I_j \cap S_{i^*} = \{v_j\}\}} \Delta(u_j, S_{i^*})}{\sum_{\{I_j \mid I_j \cap S_{i^*} = \{v_j\}\}} \sum_{p \in I_j} \Delta(p, S_{i^*}) + \sum_{\{I_j \mid I_j \cap S_{i^*} \neq \{v_j\}\}} \sum_{p\in I_j} \Delta(p, S_{i^*})} \\
&\geq \frac{(\frac{n_\alpha}{10} - m_\alpha)\cdot r^2}{(\frac{n_\alpha}{10} - m_\alpha)\cdot(2\alpha^2+1)r^2 + (n_\alpha - (\frac{n_\alpha}{10} - m_\alpha))\cdot(5\alpha^2+1)r^2} \\
&\geq \frac{m_\alpha}{m_\alpha(2\alpha^2+1) + (n_\alpha - m_\alpha)(5\alpha^2+1)} \\
&\geq \frac{1}{2\alpha^2+1+20(5\alpha^2+1)} \\
&\geq \frac{1}{123\alpha^2}
\end{aligned}$$

For each $i \in [m_\alpha]$, let random variable $X_i := \mathbf{1}_{\mathcal{E}_i | \mathcal{T}_1}$. Since $X_1, \dots, X_{m_\alpha}$ are stochastically dominated by independent random variables Y_i such that $Y_i = 1$ with probability $1/(123\alpha^2)$ and zero otherwise, we can bound the probability that none of $\mathcal{E}_1, \dots, \mathcal{E}_{m_\alpha}$ happens as follows.

$$\begin{aligned}
\Pr[\neg\mathcal{E}_1 \wedge \dots \wedge \neg\mathcal{E}_{m_\alpha} | \mathcal{T}_1] &= \Pr[\sum_{i \leq m_\alpha} X_i < 1 | \mathcal{T}_1] \\
&\leq \Pr[\sum_{i\leq m_\alpha} Y_i < 1] \\
&= (1 - \frac{1}{123\alpha^2})^{m_\alpha} \leq \exp(-\frac{m_\alpha}{123\alpha^2}) \qquad (19)
\end{aligned}$$

Hence, with probability at least $1 - \exp(-\frac{m_\alpha}{123\alpha^2})$, there exists an iteration $i^* \in [n_\alpha+1, n_\alpha+m_\alpha]$ and block I_j^* such that $I_j^* \cap S_{i^*} = \{v_j, u_j\}$. Next, we show that with constant probability no more points is picked from I_j^* in the remaining iterations of the seeding phase k-means++. Let $\mathcal{T}_2$ denote the event that there exists (I_j^*, i^*) such that $I_j^* \cap S_{i^*} = \{v_j, u_j\}$; $\mathcal{T}_2 := \mathcal{E}_1 \vee \dots \vee \mathcal{E}_{m_\alpha}$. For any $i \in [1, n_\alpha + m_\alpha - i^*]$, let $\mathcal{F}_i$ denote the event that in iteration $i^* + i$, k-means++ picks another point from I_j^*.

$$\begin{aligned}
\Pr[\mathcal{F}_i | \mathcal{T}_1 \wedge \mathcal{T}_2] &< \frac{2\alpha^2 r^2}{\sum_{\{I_j \mid I_j \cap S_{i-1} = \{v_j\}\}} \sum_{p \in I_j} \Delta(p, S_{i-1})} \\
&\leq \frac{2\alpha^2 r^2}{(\frac{n_\alpha}{10} - m_\alpha)\cdot(2\alpha^2+1)r^2} = \frac{2}{3m_\alpha} \qquad (20)
\end{aligned}$$

Let $t_\alpha := n_\alpha + m_\alpha - i^*$ and define $\mathcal{T}_3$ to denote the event that none of $\mathcal{F}_1, \dots, \mathcal{F}_{t_\alpha}$ happens. For each $i \in [t_\alpha]$, let random variable $\tilde{X}_i := \mathbf{1}_{\mathcal{F}_i | \mathcal{T}_1 \wedge \mathcal{T}_2}$. Since $\tilde{X}_1, \dots, \tilde{X}_{m_\alpha}$ are stochastically dominated by independent random variables $\tilde{Y}_i$ such that $\tilde{Y}_i = 1$ with probability $1/(123\alpha^2)$ and zero otherwise, we can lower bound the probability that event $\mathcal{T}_3$ holds as follows.

$$\begin{aligned}
\Pr[\mathcal{T}_3 | \mathcal{T}_1 \wedge \mathcal{T}_2] = \Pr[\neg\mathcal{F}_1 \wedge \dots \wedge \neg\mathcal{F}_{t_\alpha} | \mathcal{T}_1 \wedge \mathcal{T}_2] &\geq \Pr[\sum_{i \leq t_\alpha} \tilde{Y}_i < 1 | \mathcal{T}_1 \wedge \mathcal{T}_2] \\
&\geq (1 - \frac{2}{3m_\alpha})^{t_\alpha} && \triangleright \text{ by Eq. (20)} \\
&\geq (1 - \frac{2}{3m_\alpha}) \cdot \exp(-\frac{2}{3}) && \triangleright\ t_\alpha \leq m_\alpha \qquad (21)
\end{aligned}$$

Hence, Eq. (19) and Eq. (21) together imply that the probability that there exists no block I_j^* such that $|I_j^* \cap \mathcal{S}| = \{v_j, u_j\}$ is at most

$$\begin{aligned}
\Pr[\neg\mathcal{T}_1] + \Pr[\neg\mathcal{T}_2 | \mathcal{T}_1] + \Pr[\neg\mathcal{T}_3 | \mathcal{T}_1 \wedge \mathcal{T}_2] &\leq 0.02 + e^{-\frac{m_\alpha}{123\alpha^2}} + 1 - (1 - \frac{2}{3m_\alpha})e^{-\frac{2}{3}} \\
&\leq 0.02 + e^{-\frac{m_\alpha}{123\alpha^2}} + \frac{1}{2} + \frac{1}{3m_\alpha} \quad \triangleright\ m_\alpha \geq 240\alpha^2 \\
&\leq 0.02 + 0.145 + 0.5 + 0.005 < 0.67
\end{aligned}$$

Thus, with probability at least 0.33, at the end of the seeding phase, there exists a block I_j^* such that $I_j^* \cap \mathcal{S} = \{v_j, u_j\}$. $\square$

Now, we are ready to prove Theorem 16.

Proof of Theorem 16. Note that Lemma 3 together with Lemma 4 implies that with constant probability, at the end of the seeding phase, at least one point is picked from each block in $\mathcal{I}_\alpha$ and there exist a block I_j such that $I_j \cap S = \{v_j, u_j\}$.

Given the structure of $\mathcal{I}_\alpha$, after each step of the Lloyd's heuristic, no points from two different blocks will be in the same cluster. Furthermore, by Claim 15, the clusters of I_j which are initially $\{z_j, z_j', v_j\}, \{u_j\}$ remain unchanged in the final solution of k-means++. However, such clustering is violating the requirement of IP-stable clustering for v_j by a factor of α.

Thus, with constant probability, the solution of k-means++ on $(\mathcal{I}_\alpha, k_\alpha)$ violates the requirement of IP-stable clustering by a factor of at least α. $\square$

E.2. Hard Instances for k-center

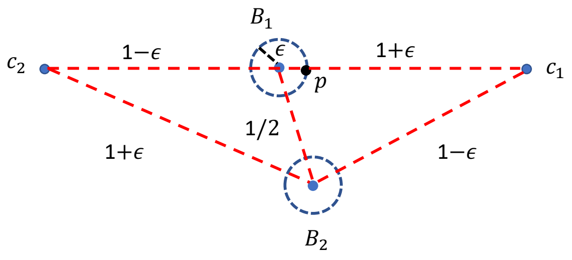

Figure 7. A hard instance for k-center.

Figure 7 shows a hard instance for k-center. Each of the balls B_1 and B_2 have radius ϵ. The center of B_1 has a distance of $1+\epsilon$ from c_1, and a distance of $1-\epsilon$ from c_2. The center of B_2 has a distance of $1-\epsilon$ from c_1, and a distance of $1+\epsilon$ from c_2. The nodes of the graph are c_1, c_2, n points on the surface of B_1, and n points on the surface of B_2.

Consider k-center using the greedy strategy by (Gonzalez, 1985). Suppose $k = 2$, and the first center picked is c_1. The furthest node from c_1 is c_2, and hence it is the second center picked. Since all the points get assigned to the closest center, all the points on B_1 get assigned to c_2, except the point p that has the same distance from c_1 and c_2 and gets assigned to c_1. All the points on B_2 get assigned to c_1. We show such a clustering is unstable for p within a factor of $n/8$. The average distance of p to the nodes in its own cluster is at least $\frac{n(1/2-2\epsilon)+1}{n+1}$. The average distance of p to the nodes assigned to c_2 is at most $\frac{2\epsilon(n-1)+1}{n}$. For $\epsilon \le 1/2n$, the instability for p is at least within a multiplicative factor of $n/8$. By setting $n > 8\alpha$, p is not t-approximately IP-stable for any value $t < \alpha$.

E.3. Hard Instances for Single Linkage

Figure 8 shows a bad example for single linkage clustering when $k = 2$. A possible implementation of single linkage first merges v_2 and v_3. Next, it merges v_4 and $\{v_2, v_3\}$, and repeatedly at each iteration i adds v_{i+2} to the set $\{v_2, v_3, \ldots, v_{i+1}\}$. When there are only two clusters $\{v_2, v_3, \ldots, v_n\}$ and v_1 left, the algorithm terminates. By setting $\epsilon = 1/2$, v_2 gets unstable by a factor of $(n-1)/4$. By setting $(n-1)/4 > \alpha$, v_2 is not t-approximately IP-stable for any value of $t < \alpha$.

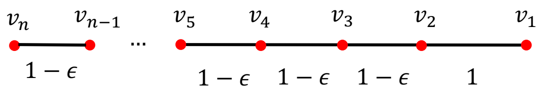

Figure 8. A hard instance for single linkage clustering.

F. Dynamic Programming Approach for Solving (3)

In the following, we propose an efficient DP approach to find a solution to (3). Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ with $x_1 \leq \ldots \leq x_n$. Our approach builds a table $T \in (\mathbb{N} \cup \{\infty\})^{n \times n \times k}$ with

$$T(i,j,l) = \min_{(C_1,\ldots,C_l) \in \mathcal{H}_{i,j,l}} \|(|C_1| - t_1, \ldots, |C_l| - t_l)\|_p^p \tag{22}$$

for $i \in [n]$, $j \in [n]$, $l \in [k]$, where

$$\mathcal{H}_{i,j,l} = \big\{\mathcal{C} = (C_1, \ldots, C_l) : \ \mathcal{C} \text{ is a stable } l\text{-clustering of } \{x_1, \ldots, x_i\} \text{ with } l \text{ non-empty} \\ \text{contiguous clusters such that the right-most cluster } C_l \text{ contains exactly } j \text{ points}\big\}$$

and $T(i,j,l) = \infty$ if $\mathcal{H}_{i,j,l} = \emptyset$. Here, we consider the case $p \neq \infty$. The modifications of our approach to the case $p = \infty$ are minimal and are described later.

The optimal value of (3) is given by $\min_{j \in [n]} T(n,j,k)^{1/p}$. Below, we will describe how to use the table T to compute an IP-stable k-clustering solving (3). First, we explain how to build T. We have, for $i, j \in [n]$,

$$T(i,j,1) = \begin{cases} |i - t_1|^p, & j = i, \\ \infty, & j \neq i \end{cases}, \qquad T(i,j,i) = \begin{cases} \sum_{s=1}^{i} |1 - t_s|^p, & j = 1, \\ \infty, & j \neq 1 \end{cases}, \\ T(i,j,l) = \infty, \quad j + l - 1 > i, \tag{23}$$

and the recurrence relation, for $l > 1$ and $j + l - 1 \leq i$,

$$T(i,j,l) = |j - t_l|^p + \min \Bigg\{ T(i-j, s, l-1) : s \in [i - j - (l-2)], \frac{\sum_{f=1}^{s-1} |x_{i-j} - x_{i-j-f}|}{s-1} \leq \\ \frac{\sum_{f=1}^{j} |x_{i-j} - x_{i-j+f}|}{j}, \frac{\sum_{f=2}^{j} |x_{i-j+1} - x_{i-j+f}|}{j-1} \leq \frac{\sum_{f=0}^{s-1} |x_{i-j+1} - x_{i-j-f}|}{s} \Bigg\}, \tag{24}$$

where we use the convention that $\frac{0}{0} = 0$ for the fractions on the left sides of the inequalities. First, we explain relation (24). Before that we need to show the following lemma holds:

Lemma 5 (Stable boundary points imply stable clustering)**.** *Let $\mathcal{C} = (C_1, \ldots, C_k)$ be a k-clustering of $\mathcal{D} = \{x_1, \ldots, x_n\}$, where $x_1 \leq x_2 \leq \ldots \leq x_n$, with contiguous clusters $C_1 = \{x_1, \ldots, x_{i_1}\}, C_2 = \{x_{i_1+1}, \ldots, x_{i_2}\}, \ldots, C_k = \{x_{i_{k-1}+1}, \ldots, x_n\}$, for some $1 \leq i_1 < \ldots < i_{k-1} < n$. Then $\mathcal{C}$ is IP-stable if and only if all points x_{i_l} and x_{i_l+1}, $l \in [k-1]$, are stable. Furthermore, x_{i_l} (x_{i_l+1}, resp.) is stable if and only if its average distance to the points in $C_l \setminus \{x_{i_l}\}$ ($C_{l+1} \setminus \{x_{i_l+1}\}$, resp.) is not greater than the average distance to the points in C_{l+1} (C_l, resp.).*

Proof. We assume that $\mathcal{D} = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}$ with $x_1 \leq x_2 \leq \ldots \leq x_n$ and write the Euclidean metric $d(x_i, x_j)$ between two points x_i and x_j in its usual way $|x_i - x_j|$.

If $\mathcal{C}$ is IP-stable, then all points x_{i_l} and x_{i_l+1}, $l \in [k-1]$, are stable. Conversely, let us assume that x_{i_l} and x_{i_l+1}, $l \in [k-1]$, are stable. We need to show that all points in $\mathcal{D}$ are stable. Let $\tilde{x} \in C_l = \{x_{i_{l-1}+1}, \ldots, x_{i_l}\}$ for some $l \in \{2, \ldots, k-1\}$ and $l' \in \{l+1, \ldots, k\}$. Since x_{i_l} is stable, we have

$$\frac{1}{|C_l| - 1} \sum_{y \in C_l} (x_{i_l} - y) = \frac{1}{|C_l| - 1} \sum_{y \in C_l} |x_{i_l} - y| \leq \frac{1}{|C_{l'}|} \sum_{y \in C_{l'}} |x_{i_l} - y| = \frac{1}{|C_{l'}|} \sum_{y \in C_{l'}} (y - x_{i_l})$$

and hence

$$\begin{aligned}\frac{1}{|C_l|-1}\sum_{y\in C_l}|\tilde{x}-y| &\leq \frac{1}{|C_l|-1}\sum_{y\in C_l\setminus\{\tilde{x}\}}(|\tilde{x}-x_{i_l}|+|x_{i_l}-y|)\\ &=(x_{i_l}-\tilde{x})+\frac{1}{|C_l|-1}\sum_{y\in C_l\setminus\{\tilde{x}\}}(x_{i_l}-y)\\ &\leq (x_{i_l}-\tilde{x})+\frac{1}{|C_{l'}|}\sum_{y\in C_{l'}}(y-x_{i_l})\\ &=\frac{1}{|C_{l'}|}\sum_{y\in C_{l'}}(y-\tilde{x})\\ &=\frac{1}{|C_{l'}|}\sum_{y\in C_{l'}}|\tilde{x}-y|.\end{aligned}$$

Similarly, we can show for $l'\in\{1,\ldots,l-1\}$ that

$$\frac{1}{|C_l|-1}\sum_{y\in C_l}|\tilde{x}-y|\leq\frac{1}{|C_{l'}|}\sum_{y\in C_{l'}}|\tilde{x}-y|,$$

and hence $\tilde{x}$ is stable. Similarly, we can show that all points $x_1,\ldots,x_{i_1-1}$ and $x_{i_{k-1}+2},\ldots,x_n$ are stable.

For the second claim observe that for $1\leq s\leq l-1$, the average distance of x_{i_l} to the points in C_s cannot be smaller than the average distance to the points in $C_l\setminus\{x_{i_l}\}$ and for $l+2\leq s\leq k$, the average distance of x_{i_l} to the points in C_s cannot be smaller than the average distance to the points in C_{l+1}. A similar argument proves the claim for x_{i_l+1}. □

It follows from Lemma 5 that a clustering $(C_1,\ldots,C_l)$ of $\{x_1,\ldots,x_i\}$ with contiguous clusters and $C_l=\{x_{i-j+1},\ldots,x_i\}$ is IP-stable if and only if $(C_1,\ldots,C_{l-1})$ is a stable clustering of $\{x_1,\ldots,x_{i-j}\}$ and the average distance of x_{i-j} to the points in $C_{l-1}\setminus\{x_{i-j}\}$ is not greater than the average distance to the points in C_l and the average distance of x_{i-j+1} to the points in $C_l\setminus\{x_{i-j+1}\}$ is not greater than the average distance to the points in C_{l-1}. The latter two conditions correspond to the two inequalities in (24) (when $|C_{l-1}|=s$, where s is a variable). By explicitly enforcing these two constraints, we can utilize the first condition and rather than minimizing over $\mathcal{H}_{i,j,l}$ in (25), we can minimize over both $s\in[i-j-(l-2)]$ and $\mathcal{H}_{i-j,s,l-1}$ (corresponding to minimizing over all IP-stable $(l-1)$-clusterings of $\{x_1,\ldots,x_{i-j}\}$ with non-empty contiguous clusters). It is

$$\min_{\substack{s\in[i-j-(l-2)]\\(C_1,\ldots,C_{l-1})\in\mathcal{H}_{i-j,s,l-1}}}\|(|C_1|-t_1,\ldots,|C_{l-1}|-t_{l-1})\|_p^p=\min_{s\in[i-j-(l-2)]}T(i-j,s,l-1),$$

and hence we end up with the recurrence relation (24).

It is not hard to see that using (24), we can build the table T in time $\mathcal{O}(n^3k)$. Once we have T, we can compute a solution $(C_1^*,\ldots,C_k^*)$ to (3) by specifying $|C_1^*|,\ldots,|C_k^*|$ in time $\mathcal{O}(nk)$ as follows: let $v^*=\min_{j\in[n]}T(n,j,k)$. We set $|C_k^*|=j_0$ for an arbitrary j_0 with $v^*=T(n,j_0,k)$. For $l=k-1,\ldots,2$, we then set $|C_l^*|=h_0$ for an arbitrary h_0 with (i) $T(n-\sum_{r=l+1}^k|C_r^*|,h_0,l)+\sum_{r=l+1}^k||C_r^*|-t_r|^p=v^*$, (ii) the average distance of $x_{n-\sum_{r=l+1}^k|C_r^*|}$ to the closest h_0-1 many points on its left side is not greater than the average distance to the points in C_{l+1}^*, and (iii) the average distance of $x_{n-\sum_{r=l+1}^k|C_r^*|+1}$ to the other points in C_{l+1}^* is not greater than the average distance to the closest h_0 many points on its left side. Finally, it is $|C_1^*|=n-\sum_{r=2}^k|C_r^*|$. It follows from the definition of the table T in (22) and Lemma 5 that for $l=k-1,\ldots,2$ we can always find some h_0 satisfying (i) to (iii) and that our approach yields an IP-stable k-clustering $(C_1^*,\ldots,C_k^*)$ of $\mathcal{D}$.

Hence we have shown the following theorem:

Theorem 17 (Efficient DP approach solves (3))**.** *By means of the dynamic programming approach (22) to (24) we can compute an IP-stable clustering solving (3) in running time $\mathcal{O}(n^3k)$.*

Let us first explain the recurrence relation (24): because of $\|(x_1,\dots,x_l)\|_p^p = \|(x_1,\dots,x_{l-1})\|_p^p + |x_l|^p$ and for every clustering $(C_1,\dots,C_l) \in \mathcal{H}_{i,j,l}$ it is $|C_l| = j$, we have

$$T(i,j,l) = |j-t_l|^p + \min_{(C_1,\dots,C_l)\in\mathcal{H}_{i,j,l}} \|(|C_1|-t_1,\dots,|C_{l-1}|-t_{l-1})\|_p^p. \tag{25}$$

F.1. $p = \infty$

Now we describe how to modify the dynamic programming approach of Section 5.1 to the case $p=\infty$: in this case, we replace the definition of the table T in (22) by

$$T(i,j,l) = \min_{(C_1,\dots,C_l)\in\mathcal{H}_{i,j,l}} \|(|C_1|-t_1,\dots,|C_l|-t_l)\|_\infty, \quad i\in[n], j\in[n], l\in[k],$$

and $T(i,j,l)=\infty$ if $\mathcal{H}_{i,j,l}=\emptyset$ as before. The optimal value of (3) is now given by $\min_{j\in[n]} T(n,j,k)$. Instead of (23), we have, for $i,j\in[n]$,

$$T(i,j,1) = \begin{cases} |i-t_1|, & j=i, \\ \infty, & j\neq i \end{cases}, \qquad T(i,j,i) = \begin{cases} \max_{s=1,\dots,i} |1-t_s|, & j=1, \\ \infty, & j\neq 1 \end{cases}$$

and

$$T(i,j,l)=\infty, \quad j+l-1>i,$$

and the recurrence relation (24) now becomes, for $l>1$ and $j+l-1\leq i$,

$$\begin{aligned} T(i,j,l) = \max\Bigg\{ |j-t_l|, \min\Bigg\{ & T(i-j,s,l-1) : s\in[i-j-(l-2)], \\ & \frac{1}{s-1}\sum_{f=1}^{s-1} |x_{i-j}-x_{i-j-f}| \leq \frac{1}{j}\sum_{f=1}^{j} |x_{i-j}-x_{i-j+f}|, \\ & \frac{1}{j-1}\sum_{f=2}^{j} |x_{i-j+1}-x_{i-j+f}| \leq \frac{1}{s}\sum_{f=0}^{s-1} |x_{i-j+1}-x_{i-j-f}| \Bigg\}\Bigg\}. \end{aligned}$$

Just like before, we can build the table T in time $\mathcal{O}(n^3k)$. Computing a solution $(C_1^*,\dots,C_k^*)$ to (3) also works similarly as before. The only thing that we have to change is the condition (i) on h_0 (when setting $|C_l^*|=h_0$ for $l=k-1,\dots,2$): now h_0 must satisfy

$$\max\left\{ T\left(n-\sum_{r=l+1}^{k} |C_r^*|, h_0, l\right), \max_{r=l+1,\dots,k} ||C_r^*|-t_r| \right\} = v^*$$

or equivalently

$$T\left(n-\sum_{r=l+1}^{k} |C_r^*|, h_0, l\right) \leq v^*.$$

G. Addendum to Section 6.1

In this section, we provide supplements to Section 6.1:

- In Section G.1, we present a simple example that shows that it really depends on the data set whether a group-fair clustering is IP-stable or not.

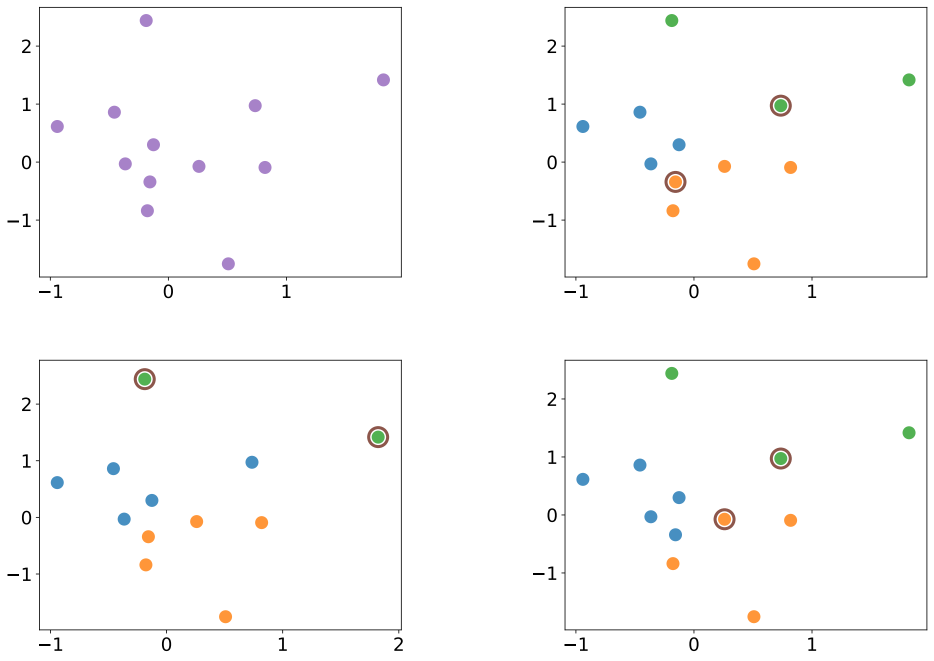

Figure 9. An example illustrating why the local search idea outlined in Section 6.1 does not work. **Top left:** 12 points in $\mathbb{R}^2$. **Top right:** A k-means clustering of the 12 points (encoded by color) with two points that are not IP-stable (surrounded by a circle). **Bottom row:** After assigning one of the two points that are not stable in the k-means clustering to its closest cluster, that point is stable. However, now some points that were initially stable are not stable anymore.

- In Section G.2, we provide an example illustrating why the local search idea of Section 6.1 does not work.
- In Section G.3, we present a heuristic approach to make linkage clustering more aligned with IP-stability.
- In Section G.4, we present the experiments of Section 6.1 on the Adult data set (Dua and Graff, 2019): we study the performance of various standard clustering algorithms as a function of the number of clusters k when the underlying metric is either the Euclidean (as in the plot of Figure 3), Manhattan or Chebyshev metric. We also study our heuristic approach of Appendix G.3 for making linkage clustering more stable: just as for the standard algorithms, we show # Uns, MaxVi, MeanVi and Co as a function of k for ordinary average / complete / single linkage clustering and their modified versions using our heuristic approach in its both variants (#U denotes the variant based on # Uns and MV the variant based on MaxVi).
- In Section G.5, we show the same set of experiments on the Drug Consumption data set (Fehrman et al., 2015). We used all 1885 records in the data set, and we used all 12 features describing a record (e.g., age, gender, or education), but did not use the information about the drug consumption of a record (this information is usually used as label when setting up a classification problem on the data set). We normalized the features to zero mean and unit variance. When running the standard clustering algorithms on the data set, we refrained from running spectral clustering since the Scikit-learn implementation occasionally was not able to do the eigenvector computations and aborted with a LinAlgError. Other than that, all results are largely consistent with the results for the Adult data set.
- In Section G.6, we show the same set of experiments on the Indian Liver Patient data set (Dua and Graff, 2019). Removing four records with missing values, we ended up with 579 records, for which we used all 11 available features (e.g., age, gender, or total proteins). We normalized the features to zero mean and unit variance. Again, all results are largely consistent with the results for the Adult data set.

G.1. Compatibility of Group Fairness and IP-Stability

By means of a simple example we want to illustrate that it really depends on the data set whether group fairness and IP-stability are compatible or at odds with each other. Here we consider the prominent group fairness notion for clustering of Chierichetti et al. (2017), which asks that in each cluster, every demographic group is approximately equally represented. Let us assume that the data set consists of the four 1-dimensional points 0, 1, 7 and 8 and the distance function d is the ordinary Euclidean metric. It is easy to see that the only IP-stable 2-clustering is $\mathcal{C} = (\{0,1\},\{7,8\})$. Now if there are two demographic groups G_1 and G_2 with $G_1 = \{0,7\}$ and $G_2 = \{1,8\}$, the clustering $\mathcal{C}$ is perfectly fair according to the notion of Chierichetti et al.. But if $G_1 = \{0,1\}$ and $G_2 = \{7,8\}$, the clustering $\mathcal{C}$ is totally unfair according to the their notion.

G.2. Why Local Search Does not Work

Figure 9 presents an example illustrating why the local search idea outlined in Section 6.1 does not work: assigning a data point that is not IP-stable to its closest cluster (so that that data point becomes stable) may cause other data points that are initially stable to not be stable anymore after the reassignment.

G.3. Heuristic Approach to Make Linkage Clustering More Stable

Linkage clustering builds a binary tree that represents a hierarchical clustering with the root of the tree corresponding to the whole data set and every node corresponding to a subset such that a parent is the union of its two children. The leaves of the tree correspond to singletons comprising one data point (e.g., Shalev-Shwartz and Ben-David, 2014, Section 22.1). If one wants to obtain a k-clustering of the data set, the output of a linkage clustering algorithm is a certain pruning of this tree.

With IP-stability being our primary goal, we propose to construct a k-clustering / a pruning of the tree as follows: starting with the two children of the root, we maintain a set of nodes that corresponds to a clustering and proceed in $k-2$ rounds. In round i, we greedily split one of the $i+1$ many nodes that we currently have into its two children such that the resulting $(i+2)$-clustering minimizes, over the $i+1$ many possible splits, # Uns (the number of non-stable datapoints; cf. the beginning of Section 6). Alternatively, we can split the node that gives rise to a minimum value of MaxVi (cf. the beginning of Section 6). Algorithm 1 provides the pseudocode of our proposed strategy.

Algorithm 1 Algorithm to greedily prune a hierarchical clustering

```
1: Input: binary tree $T$ representing a hierarchical clustering obtained from running a linkage clustering algorithm; number
   of clusters $k \in \{2,\dots,|\mathcal{D}|\}$; measure $meas \in \{\#\,\mathrm{Uns}, \mathrm{MaxVi}\}$ that one aims to optimize for
2: Output: a $k$-clustering $\mathcal{C}$
3: # Conventions:
   • for a node $v \in T$, we denote the left child of $v$ by $Left(v)$ and the right child by $Right(v)$
   • for a $j$-clustering $\mathcal{C}' = (C_1, C_2, \dots, C_j)$, a cluster $C_l$ and $A, B \subseteq C_l$ with $A \dot\cup B = C_l$ we write $\mathcal{C}'|_{C_l \hookrightarrow A,B}$ for the
     $(j+1)-$clustering that we obtain by replacing the cluster $C_l$ with two clusters $A$ and $B$ in $\mathcal{C}'$
4: Let $r$ be the root of $T$ and initialize the clustering $\mathcal{C}$ as $\mathcal{C} = (Left(r), Right(r))$
5: for $i = 1$ to $k-2$ by $1$ do
6:     Set
       $$v^\star = \operatorname*{argmin}_{v:v \text{ is a cluster in } \mathcal{C} \text{ with } |v|>1} meas(\mathcal{C}|_{v \hookrightarrow Left(v), Right(v)})$$
       and $\mathcal{C} = \mathcal{C}|_{v^\star \hookrightarrow Left(v^\star), Right(v^\star)}$
7: end for
8: return $\mathcal{C}$
```

G.4. Experiments on the Adult Data Set

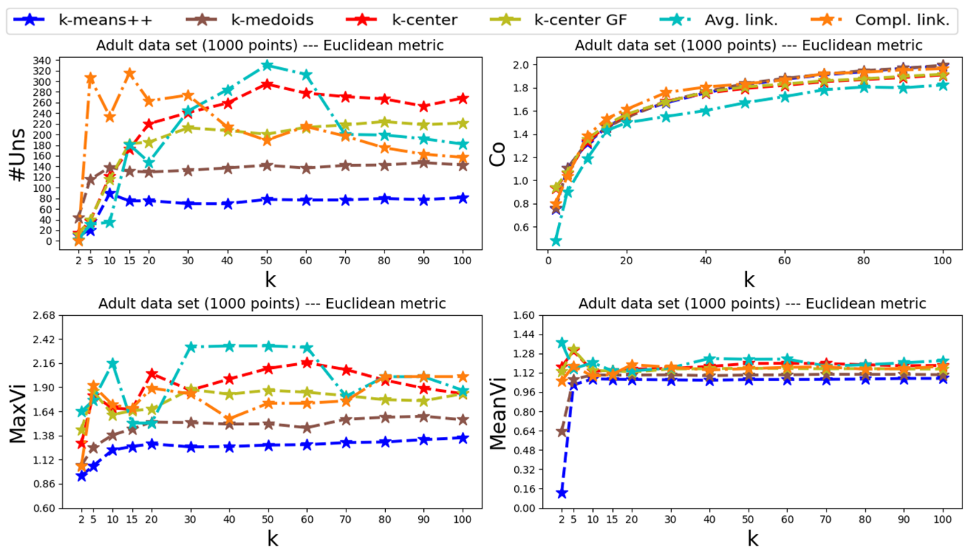

Figure 10. Adult data set — plots of Figure 3: # Uns **(top-left)**, Co **(top-right)**, MaxVi **(bottom-left)** and MeanVi **(bottom-right)** for the clusterings produced by the various standard algorithms as a function of k.

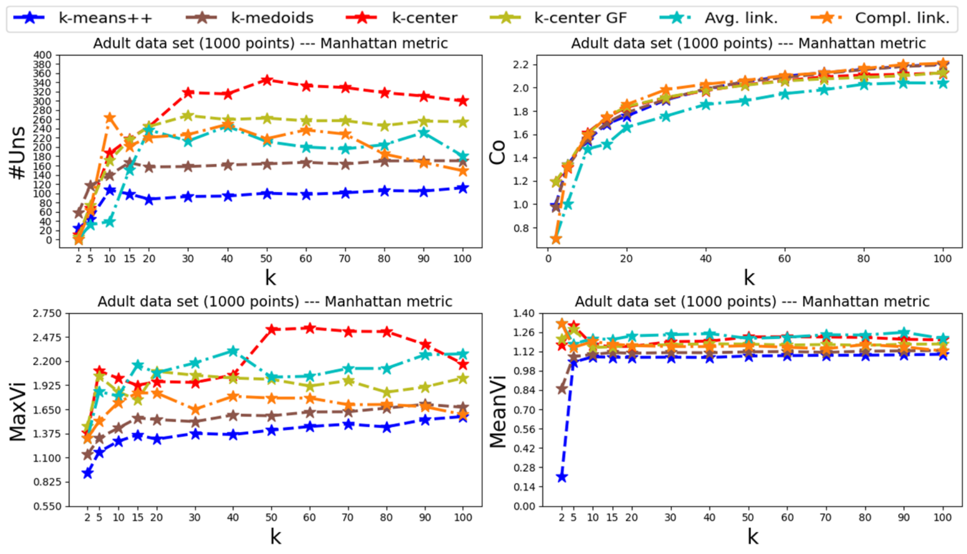

Figure 11. Adult data set — similar plots as in Figure 3, but for the Manhattan metric.

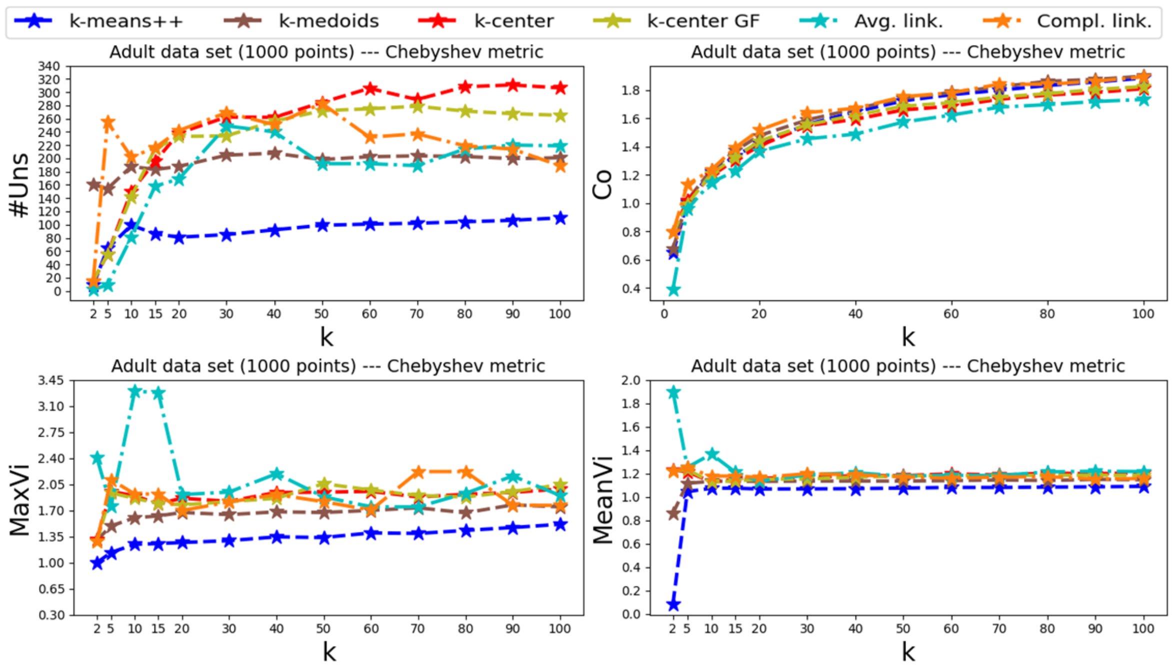

Figure 12. Adult data set — similar plots as in Figure 3, but for the Chebyshev metric.

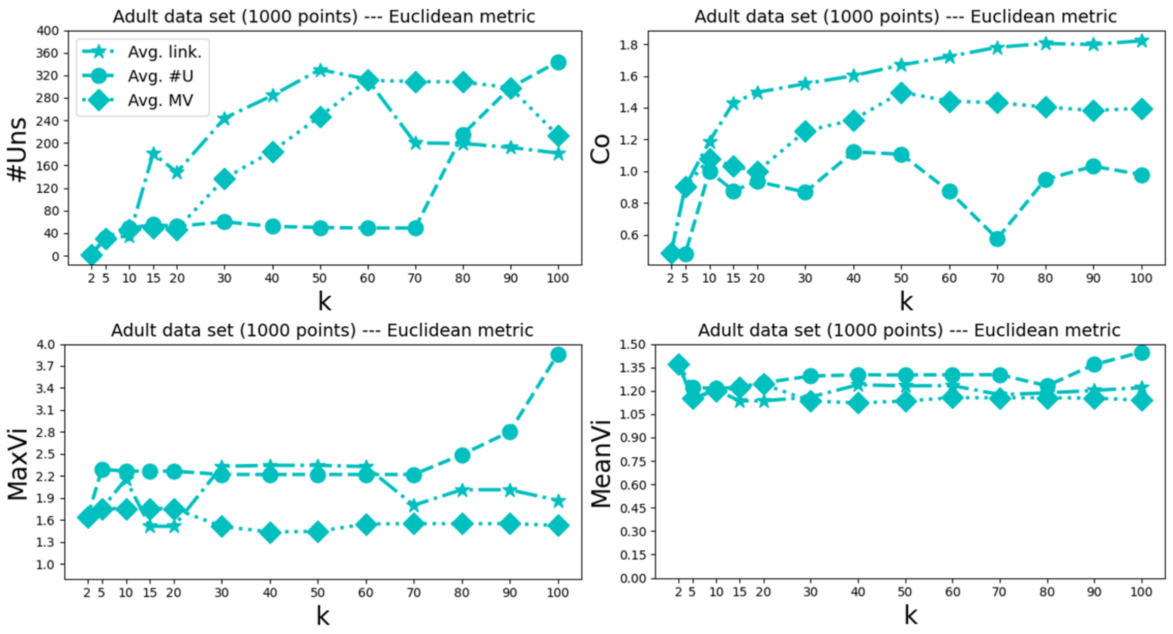

Figure 13. Adult data set with Euclidean metric: # Uns **(top-left)**, Co **(top-right)**, MaxVi **(bottom-left)**, and MeanVi **(bottom-right)** for the clusterings produced by average linkage clustering and the two variants of our heuristic of Appendix G.3 to improve it: the first (#U in the legend) greedily chooses splits as to minimize # Uns, the second (MV) as to minimize MaxVi.

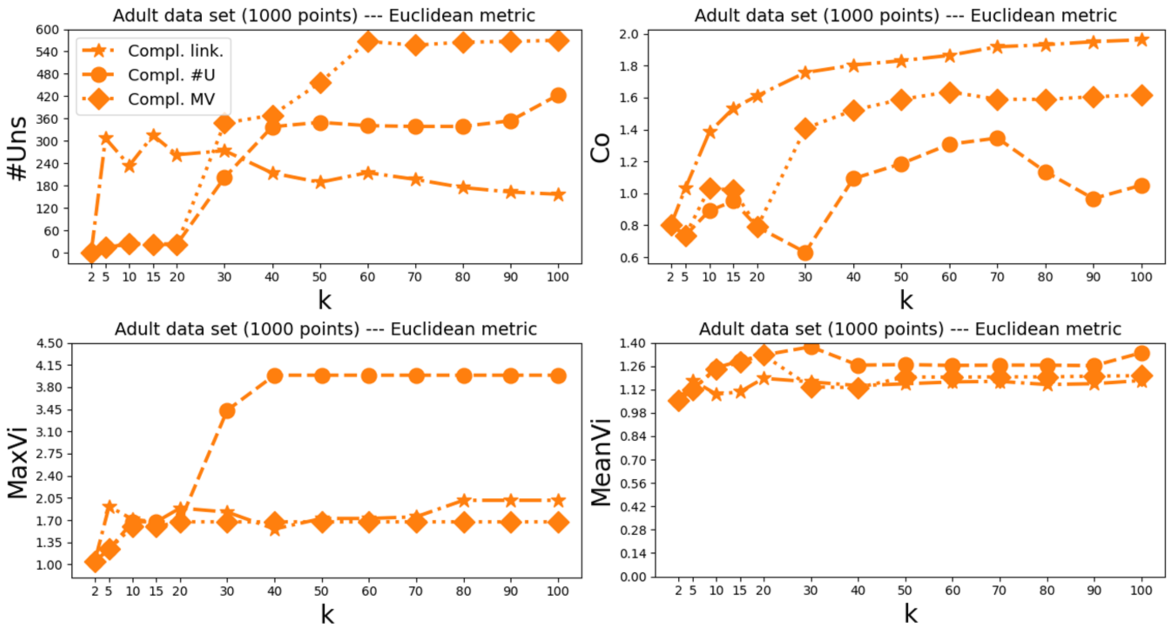

Figure 14. Adult data set with Euclidean metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

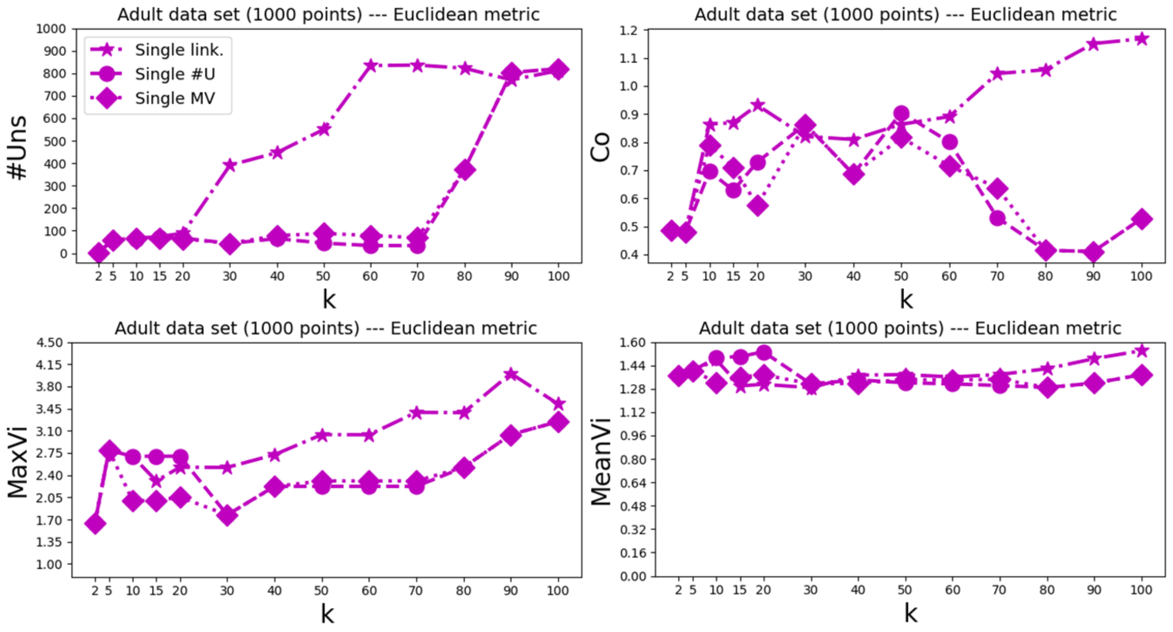

Figure 15. Adult data set with Euclidean metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

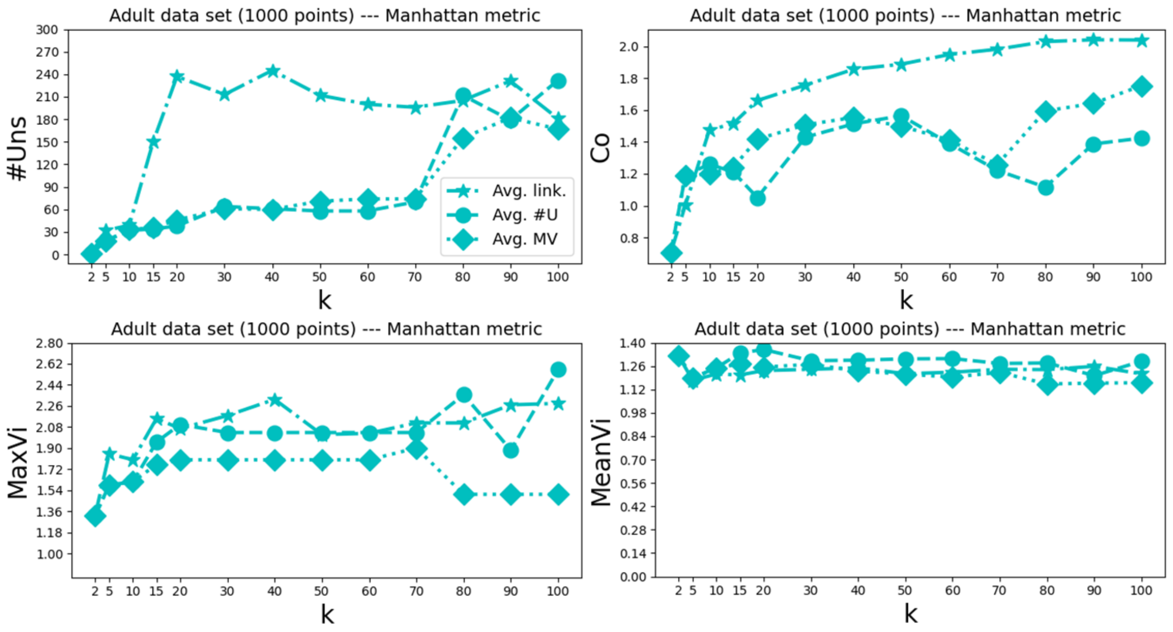

Figure 16. Adult data set with Manhattan metric: the various measures for the clusterings produced by average linkage clustering and the two variants of our heuristic approach.

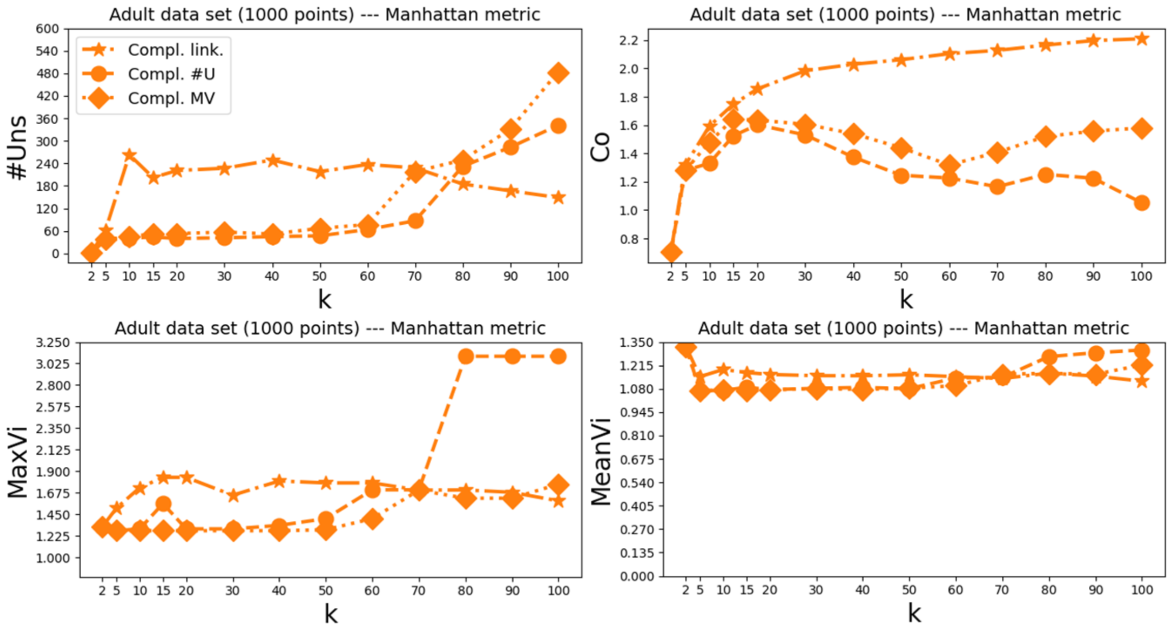

Figure 17. Adult data set with Manhattan metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

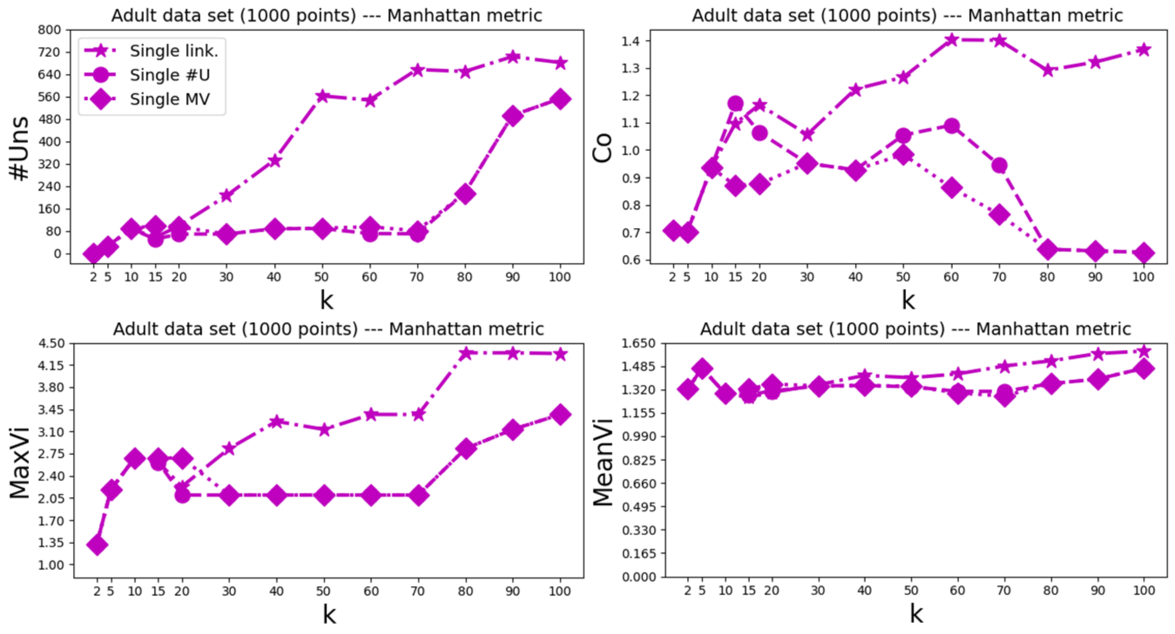

Figure 18. Adult data set with Manhattan metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

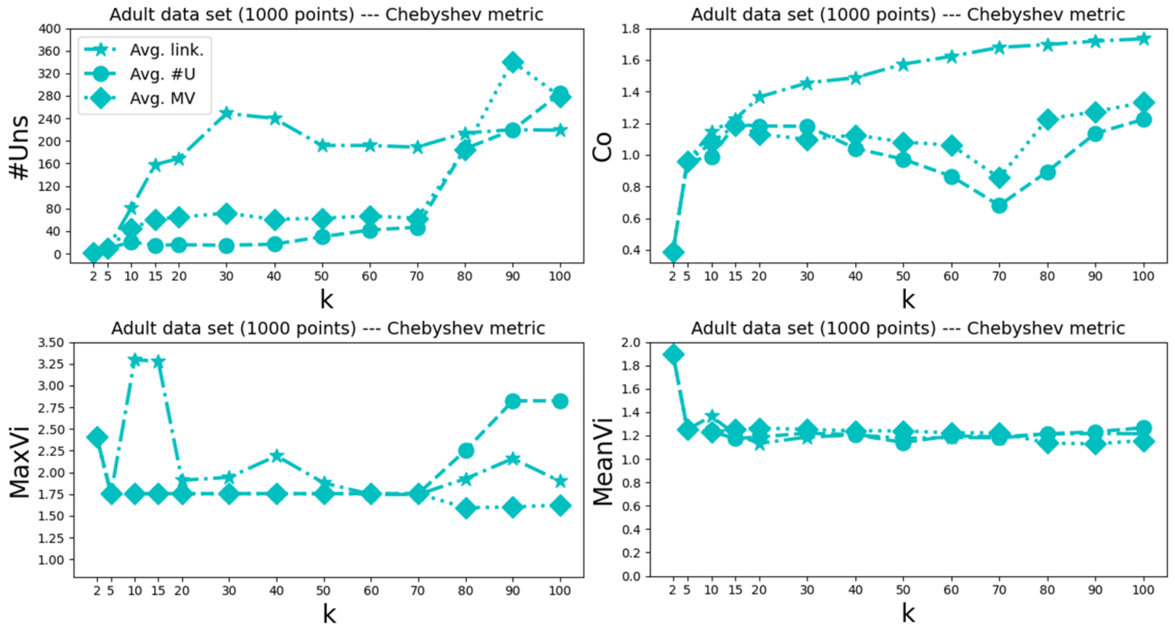

Figure 19. Adult data set with Chebyshev metric: the various measures for the clusterings produced by average linkage clustering and the two variants of our heuristic approach.

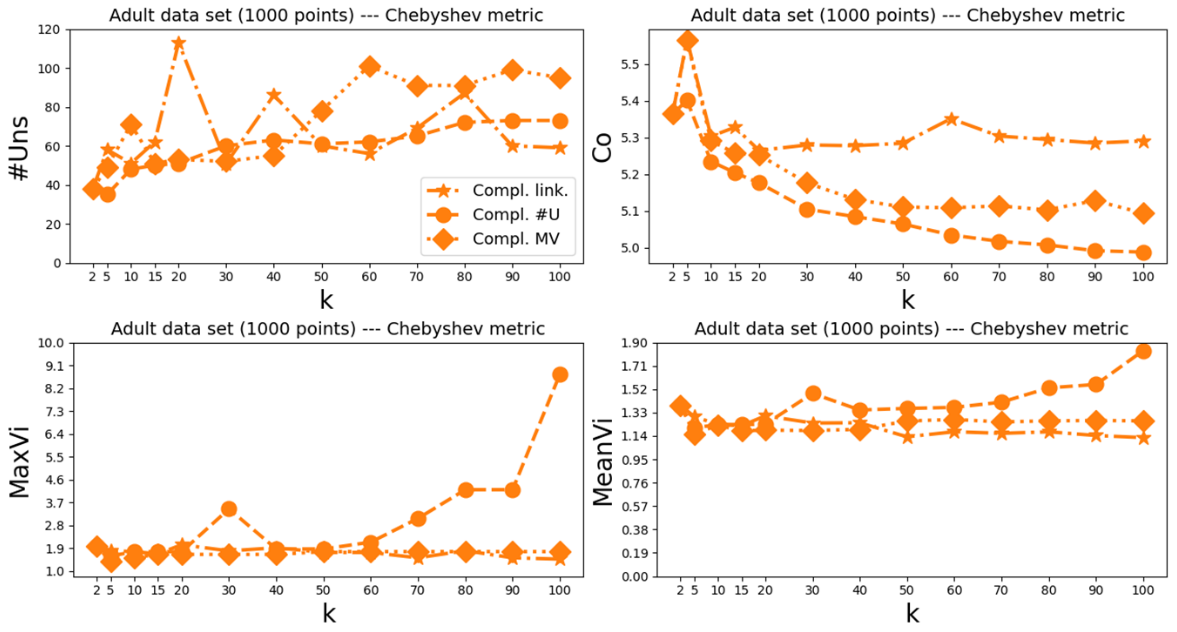

Figure 20. Adult data set with Chebyshev metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

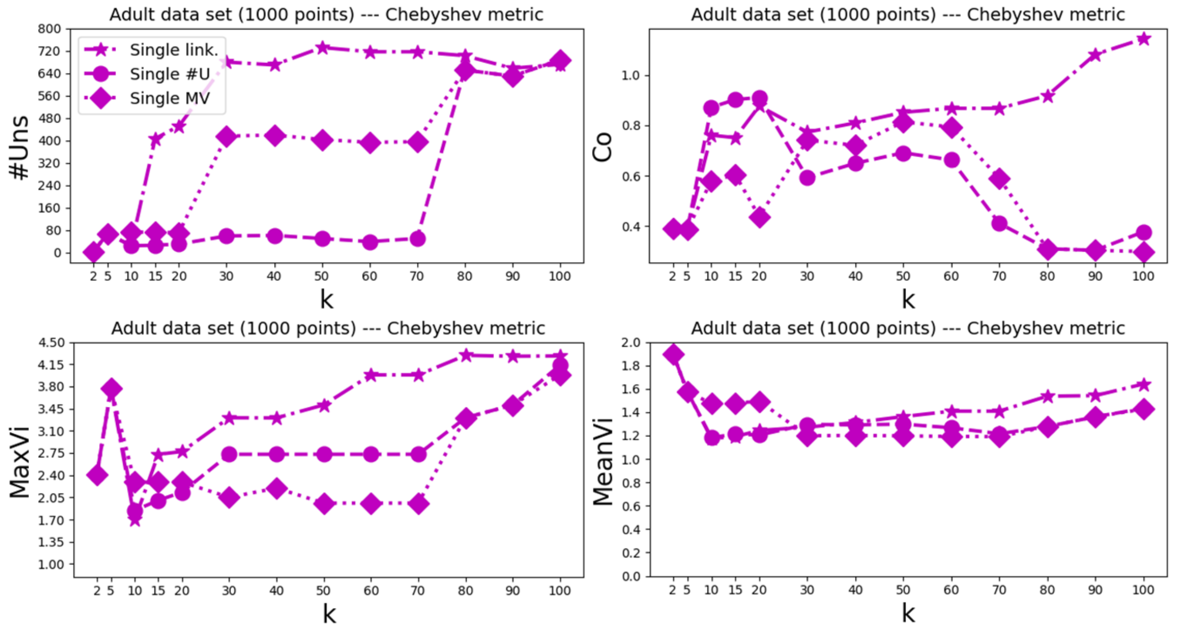

Figure 21. Adult data set with Chebyshev metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

G.5. Experiments on the Drug Consumption Data Set

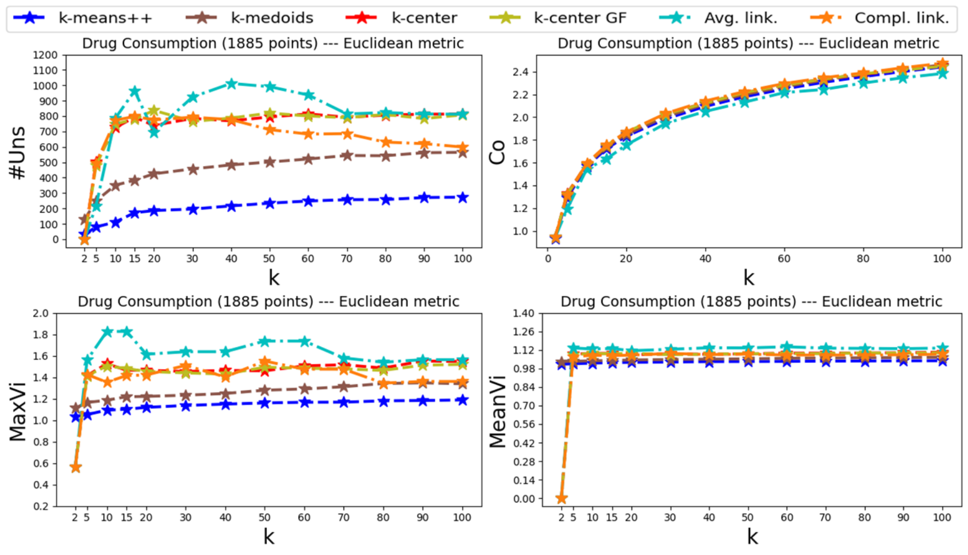

Figure 22. Drug Consumption data set with Euclidean metric: # Uns **(top-left)**, Co **(top-right)**, MaxVi **(bottom-left)** and MeanVi **(bottom-right)** for the clusterings produced by the various standard algorithms as a function of the number of clusters k. k.

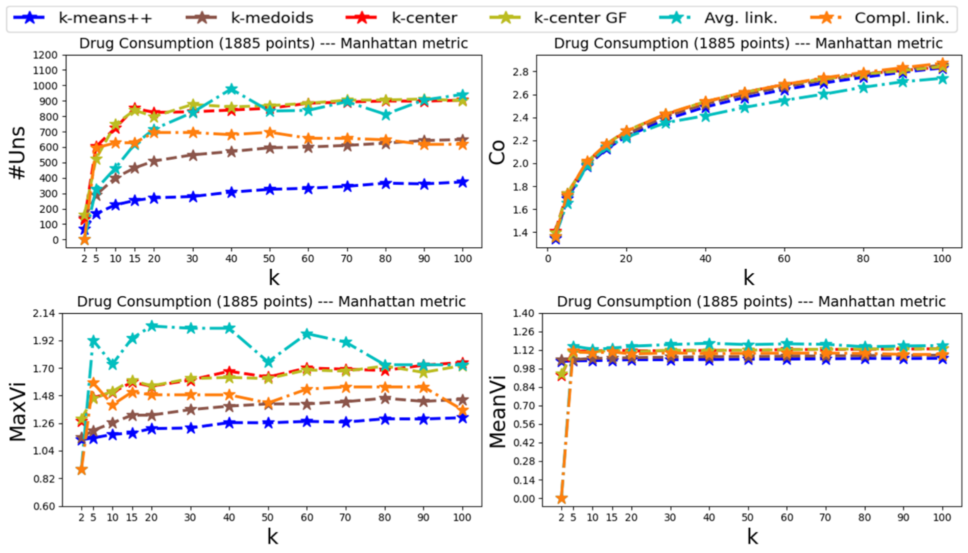

Figure 23. Drug Consumption data set — similar plots as in Figure 22, but for the Manhattan metric.

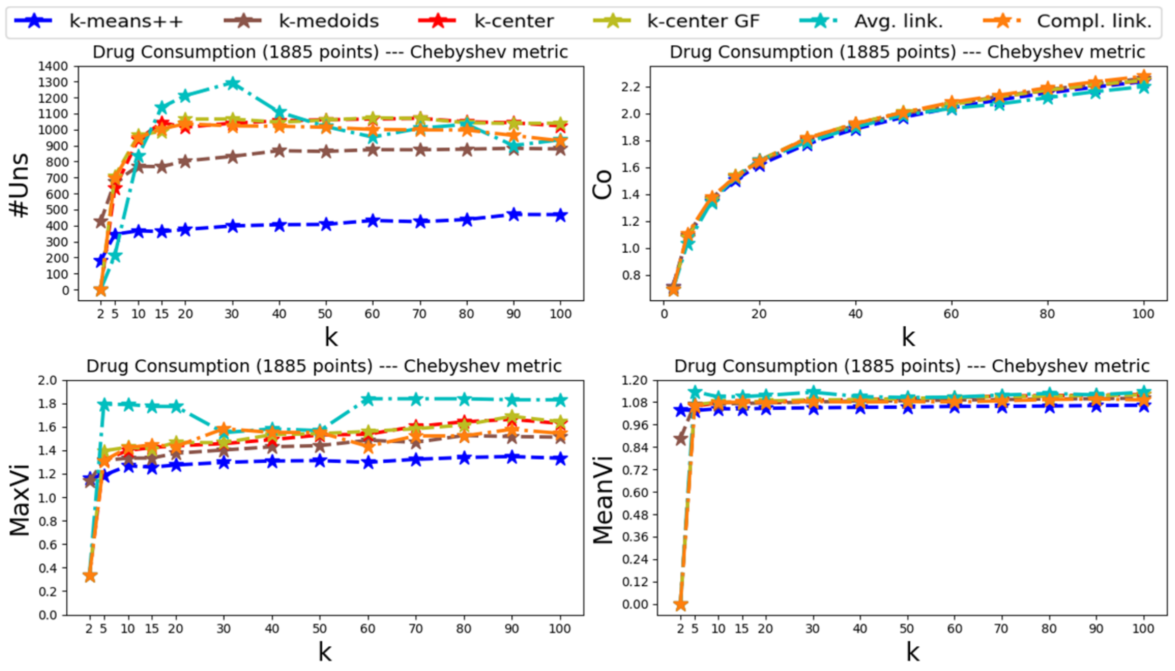

Figure 24. Drug Consumption data set — similar plots as in Figure 22, but for the Chebyshev metric.

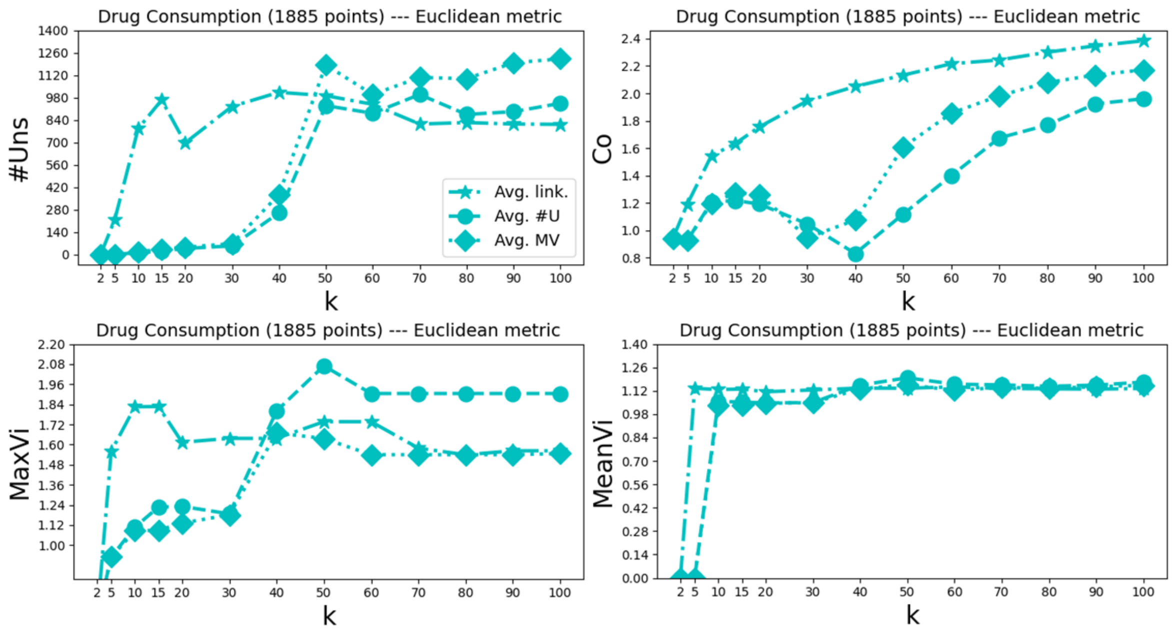

Figure 25. Drug Consumption data set with Euclidean metric: # Uns **(top-left)**, Co **(top-right)**, MaxVi **(bottom-left)**, and MeanVi **(bottom-right)** for the clusterings produced by average linkage clustering and the two variants of our heuristic of Appendix G.3 to improve it: the first (#U in the legend) greedily chooses splits as to minimize # Uns, the second (MV) as to minimize MaxVi.

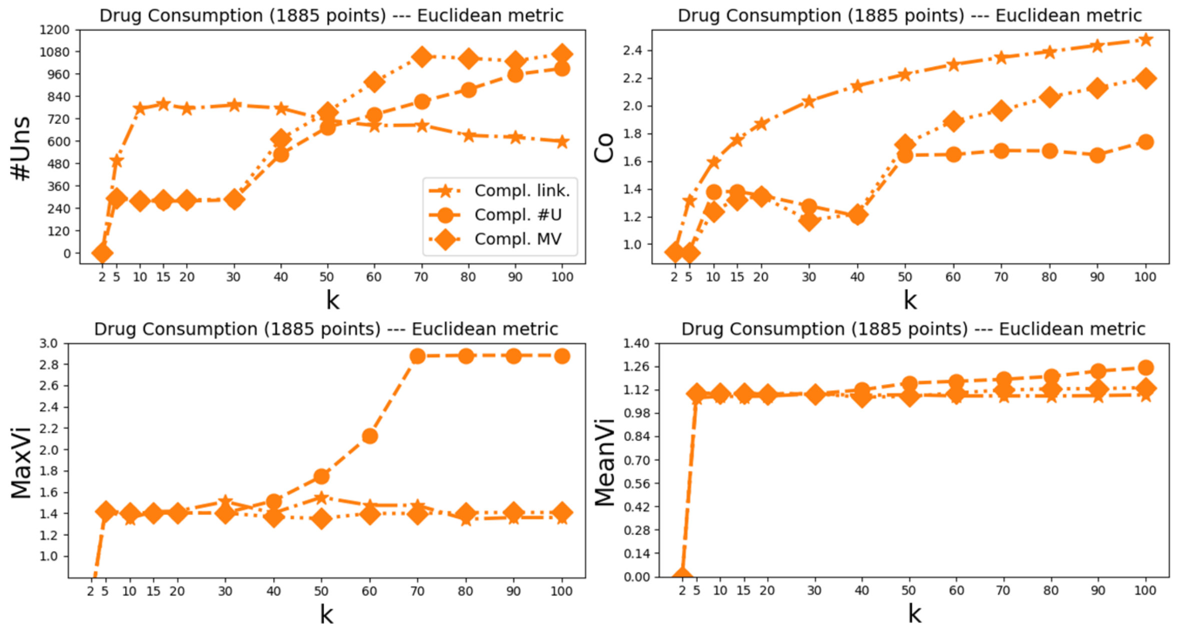

Figure 26. Drug Consumption data set with Euclidean metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

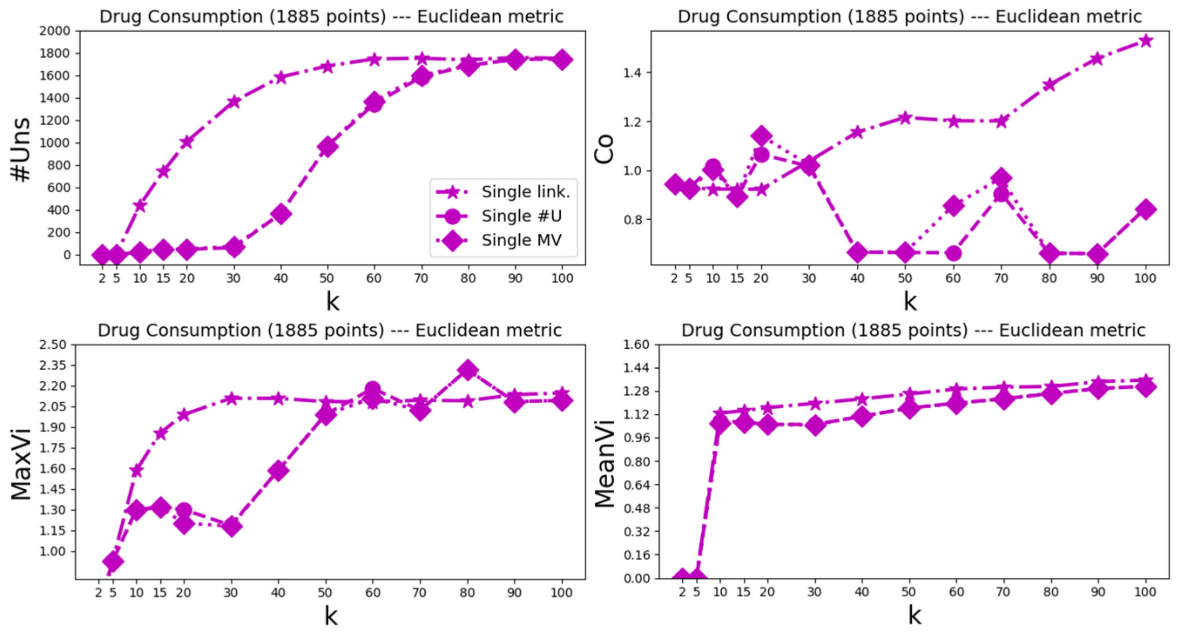

Figure 27. Drug Consumption data set with Euclidean metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

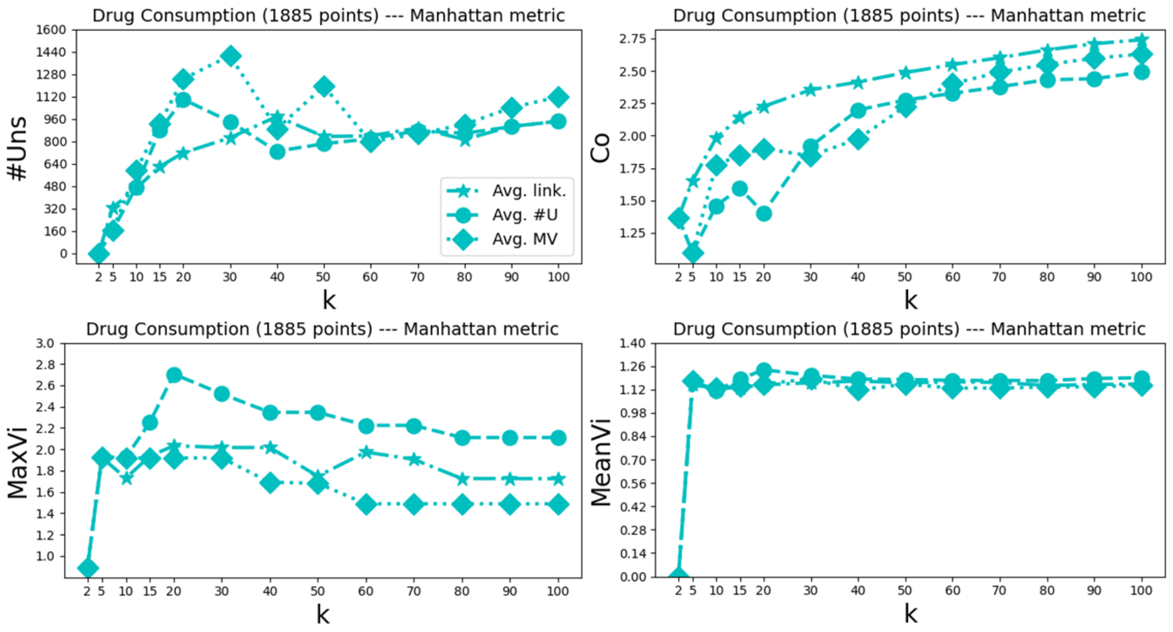

Figure 28. Drug Consumption data set with Manhattan metric: the various measures for the clusterings produced by average linkage clustering and the two variants of our heuristic approach.

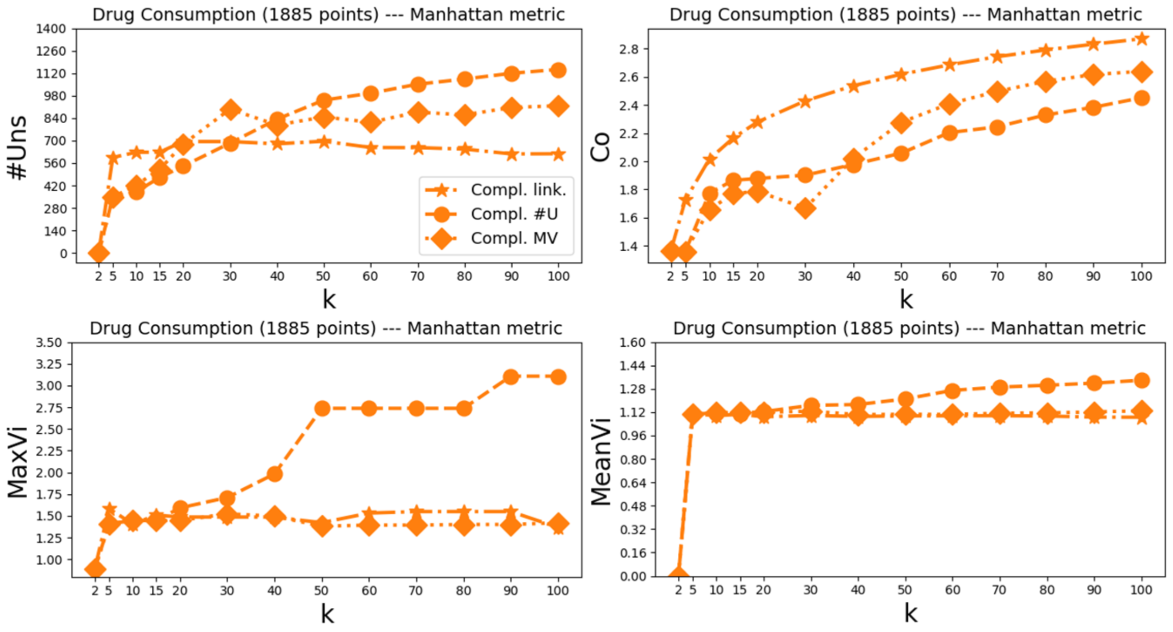

Figure 29. Drug Consumption data set with Manhattan metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

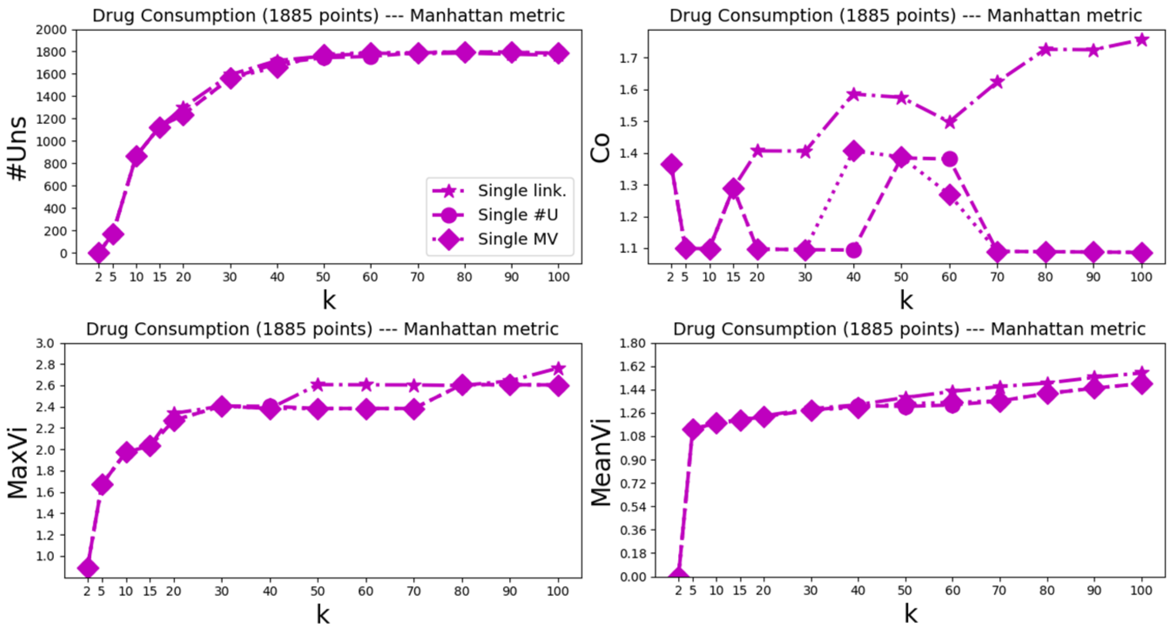

Figure 30. Drug Consumption data set with Manhattan metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

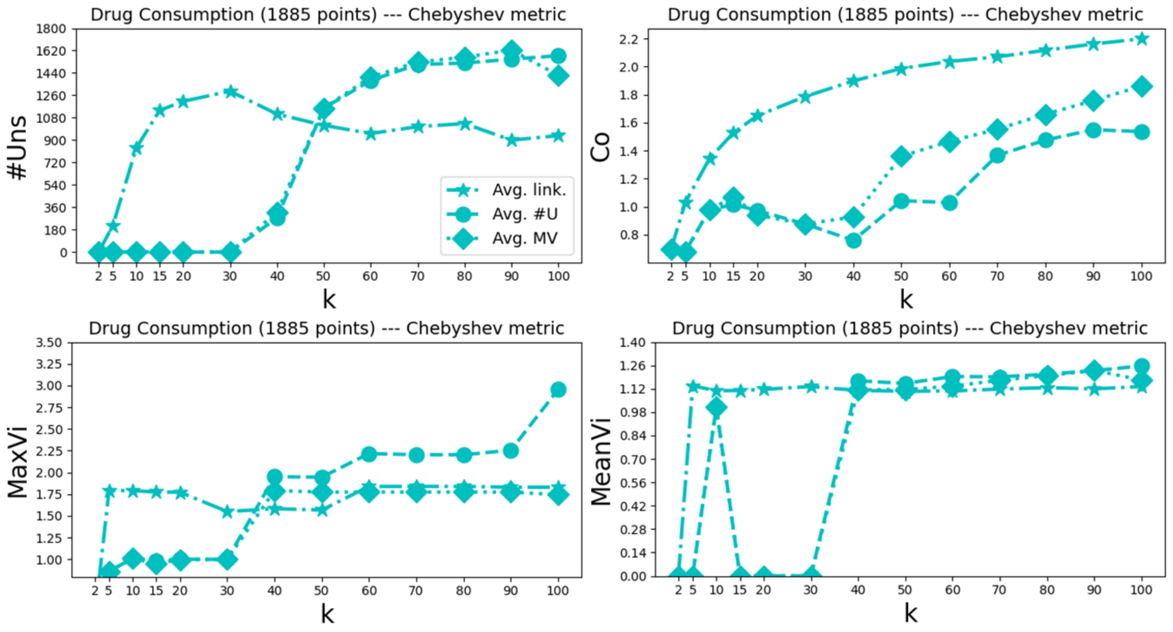

Figure 31. Drug Consumption data set with Chebyshev metric: the various measures for the clusterings produced by average linkage clustering and the two variants of our heuristic approach.

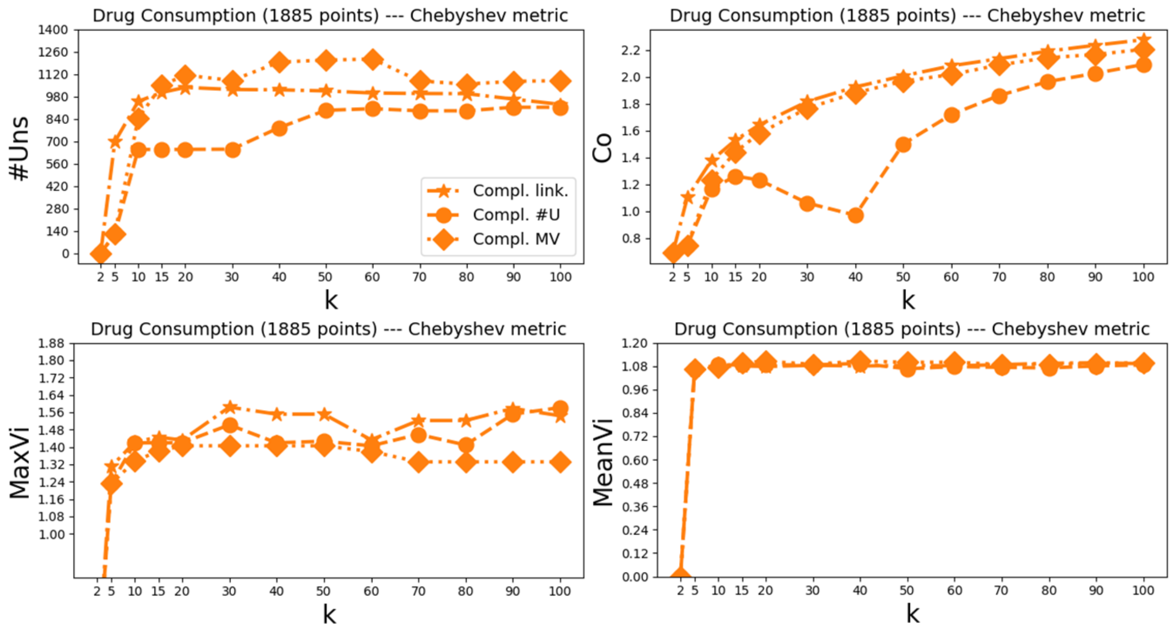

Figure 32. Drug Consumption data set with Chebyshev metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

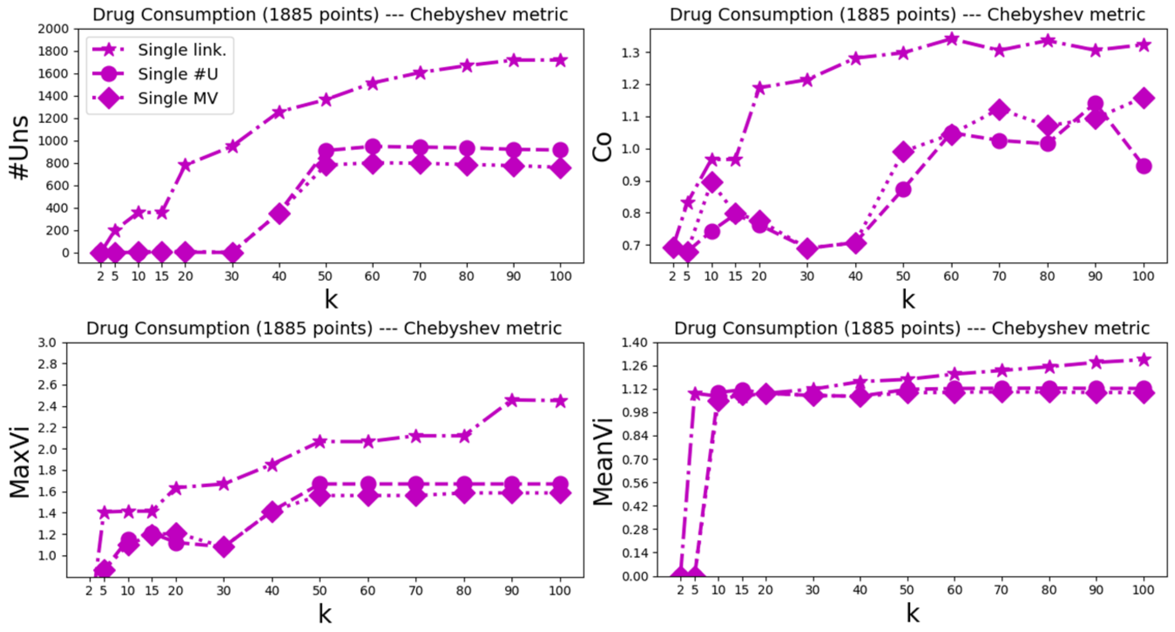

Figure 33. Drug Consumption data set with Chebyshev metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

G.6. Experiments on the Indian Liver Patient Data Set

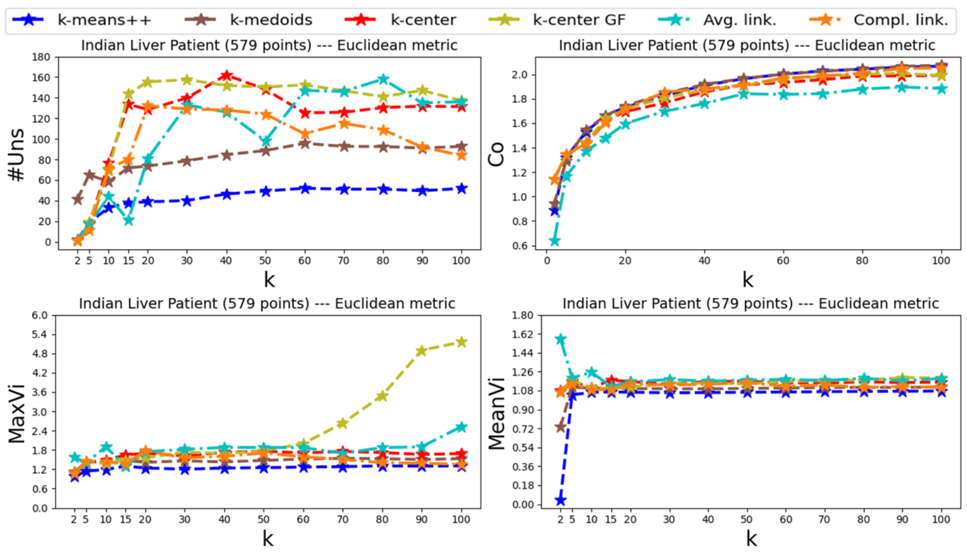

Figure 34. Indian Liver Patient data set with Euclidean metric: # Uns **(top-left)**, Co **(top-right)**, MaxVi **(bottom-left)** and MeanVi **(bottom-right)** for the clusterings produced by the various standard algorithms as a function of the number of clusters k. k.

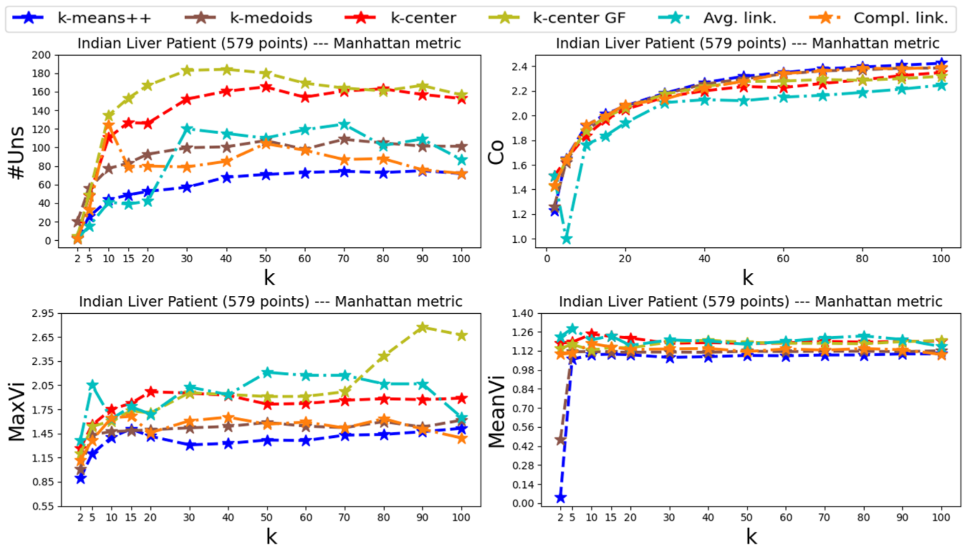

Figure 35. Indian Liver Patient data set — similar plots as in Figure 34, but for the Manhattan metric.

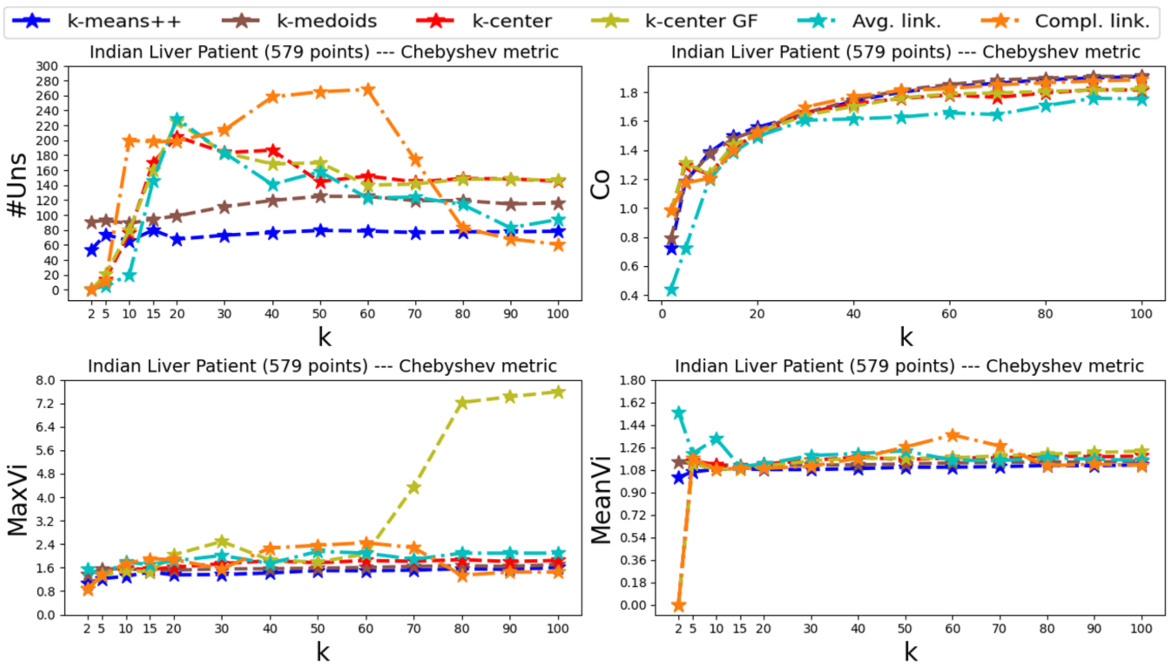

Figure 36. Indian Liver Patient data set — similar plots as in Figure 34, but for the Chebyshev metric.

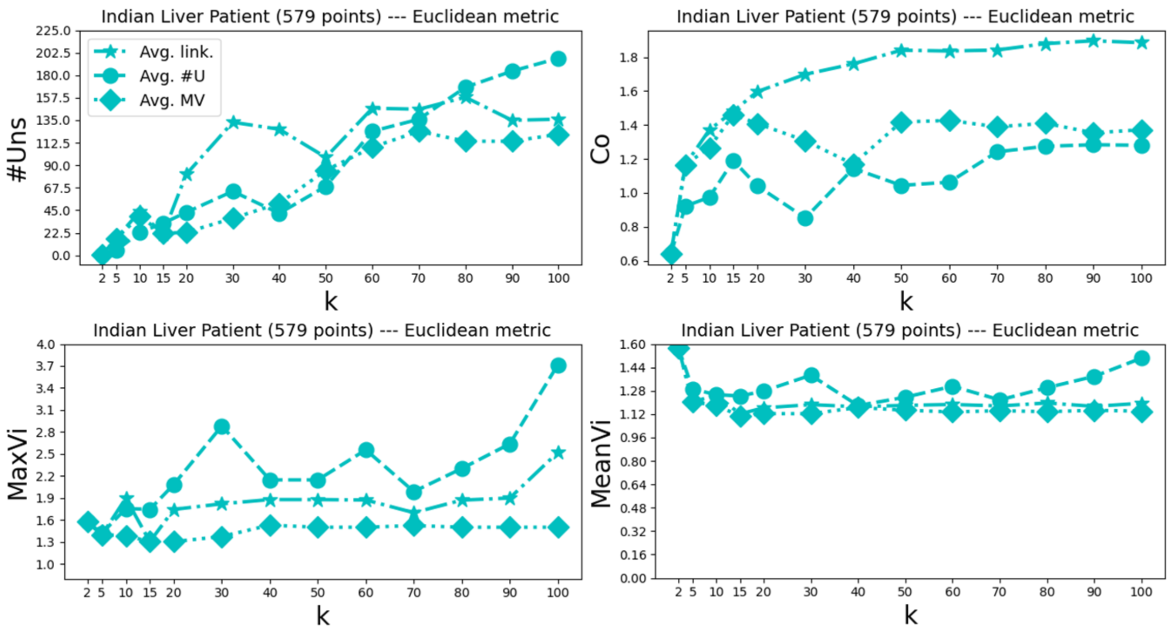

Figure 37. Indian Liver Patient data set with Euclidean metric: # Uns **(top-left)**, Co **(top-right)**, MaxVi **(bottom-left)**, and MeanVi **(bottom-right)** for the clusterings produced by average linkage clustering and the two variants of our heuristic of Appendix G.3 to improve it: the first (#U in the legend) greedily chooses splits as to minimize # Uns, the second (MV) as to minimize MaxVi.

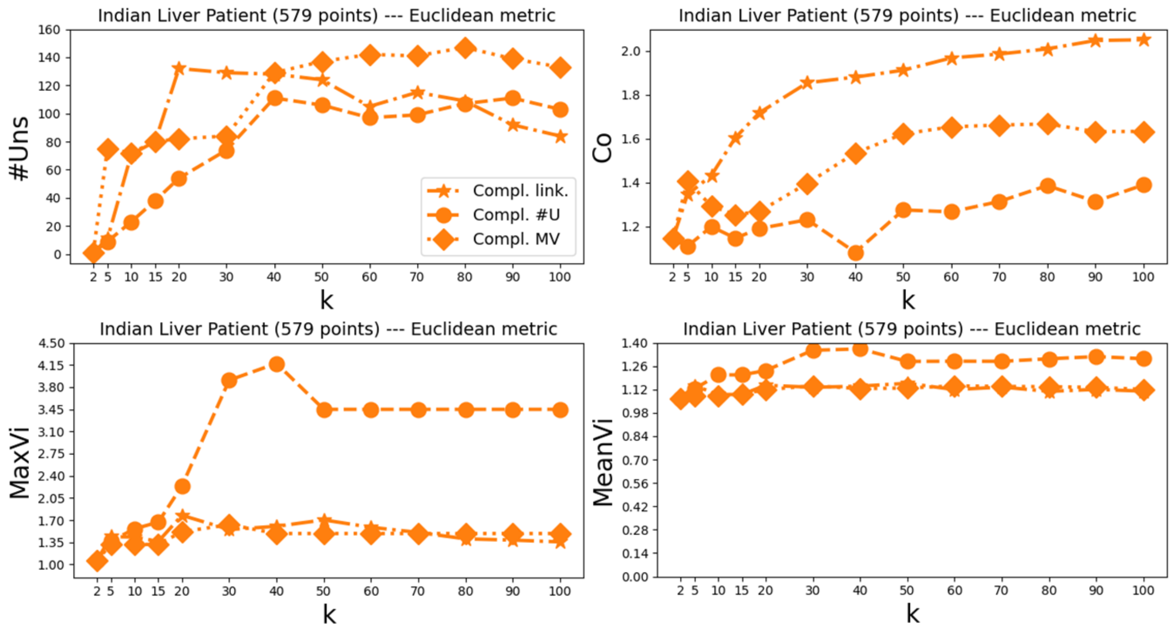

Figure 38. Indian Liver Patient data set with Euclidean metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

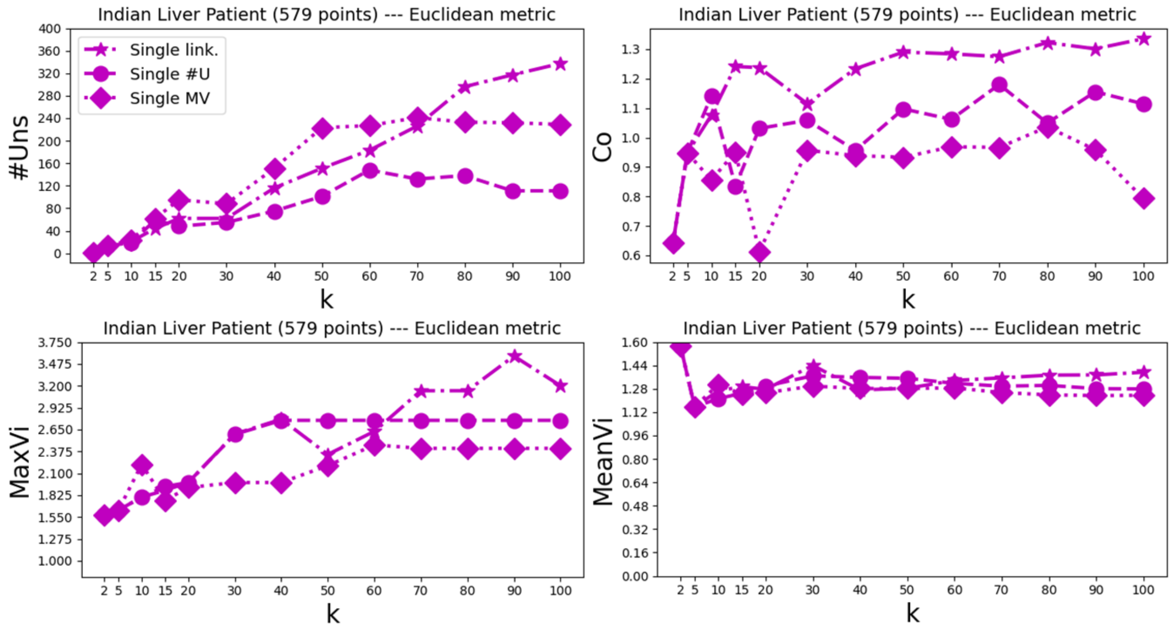

Figure 39. Indian Liver Patient data set with Euclidean metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

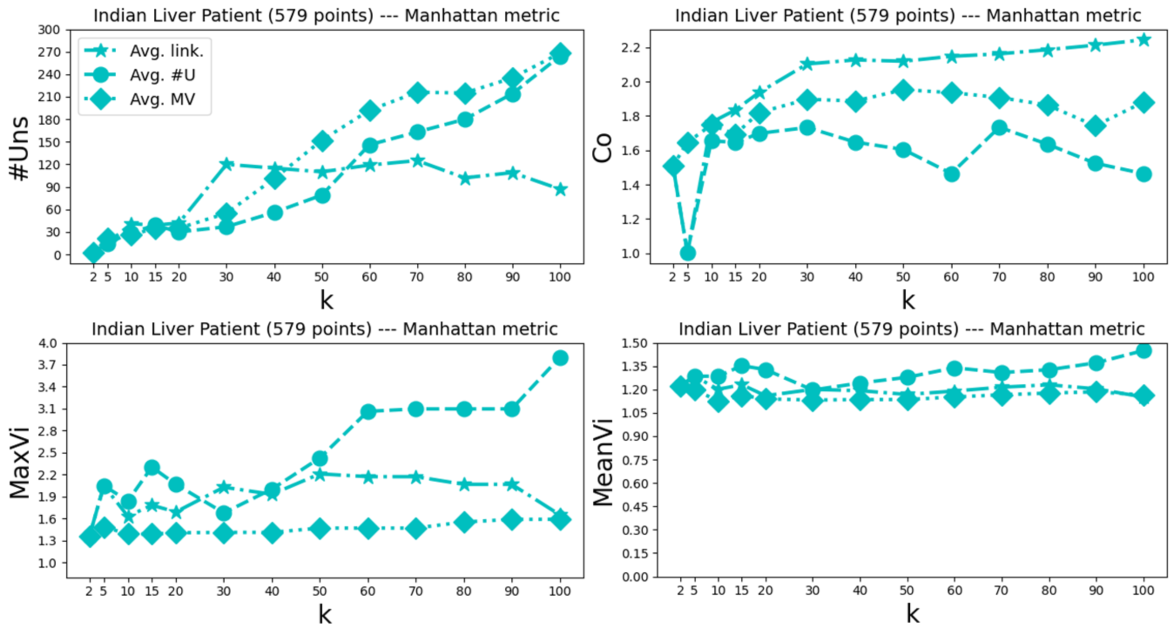

Figure 40. Indian Liver Patient data set with Manhattan metric: the various measures for the clusterings produced by average linkage clustering and the two variants of our heuristic approach.

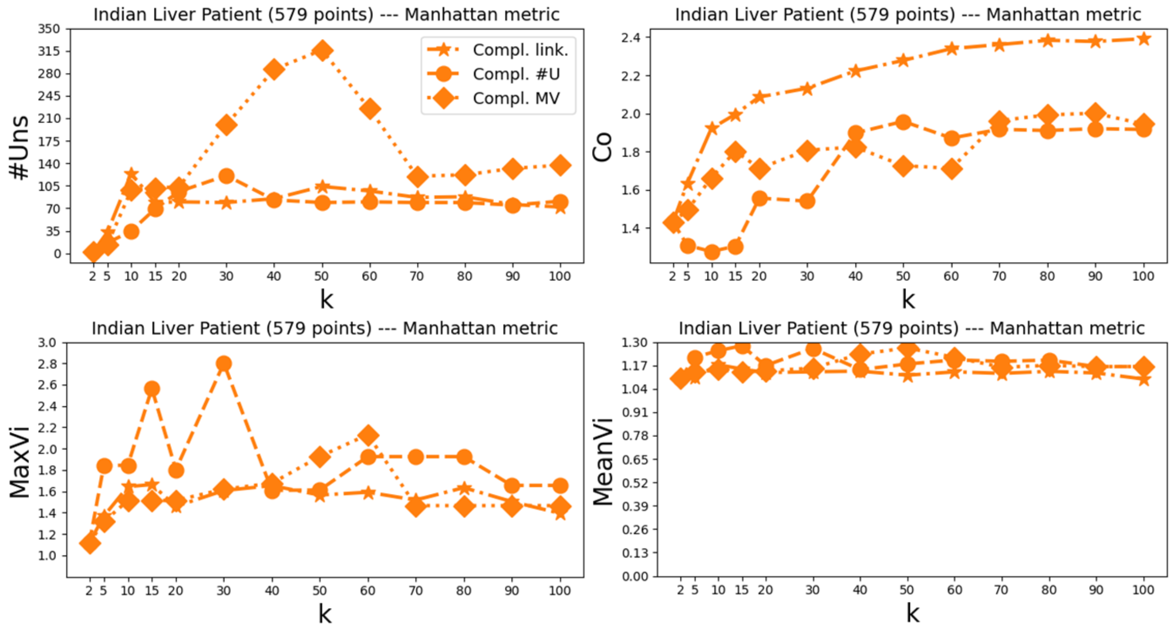

Figure 41. Indian Liver Patient data set with Manhattan metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

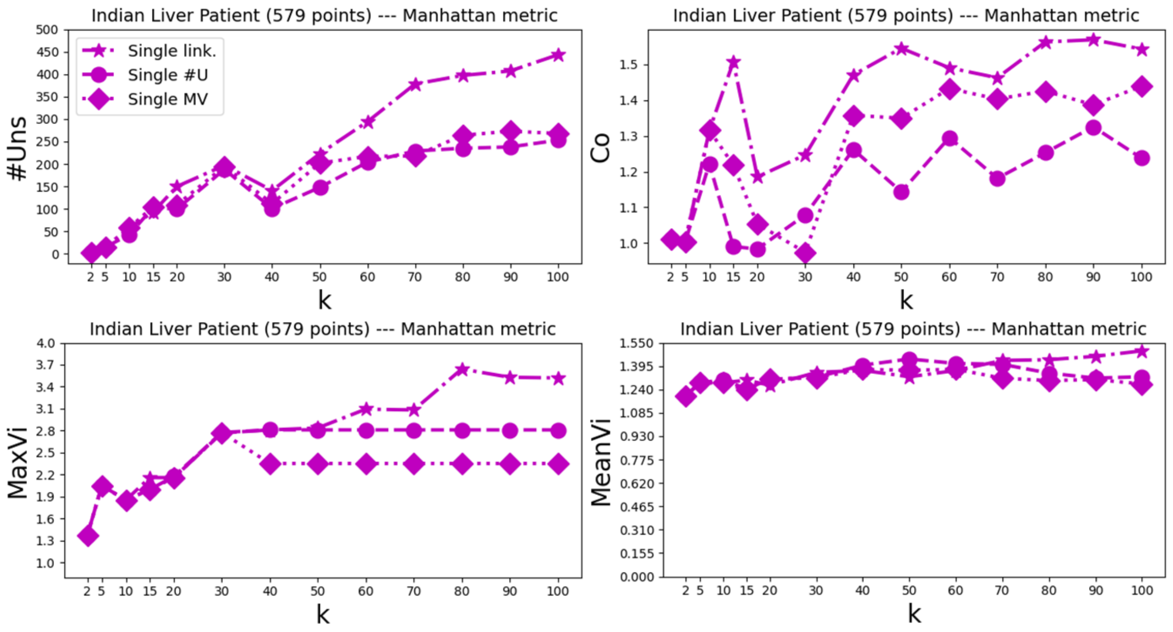

Figure 42. Indian Liver Patient data set with Manhattan metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

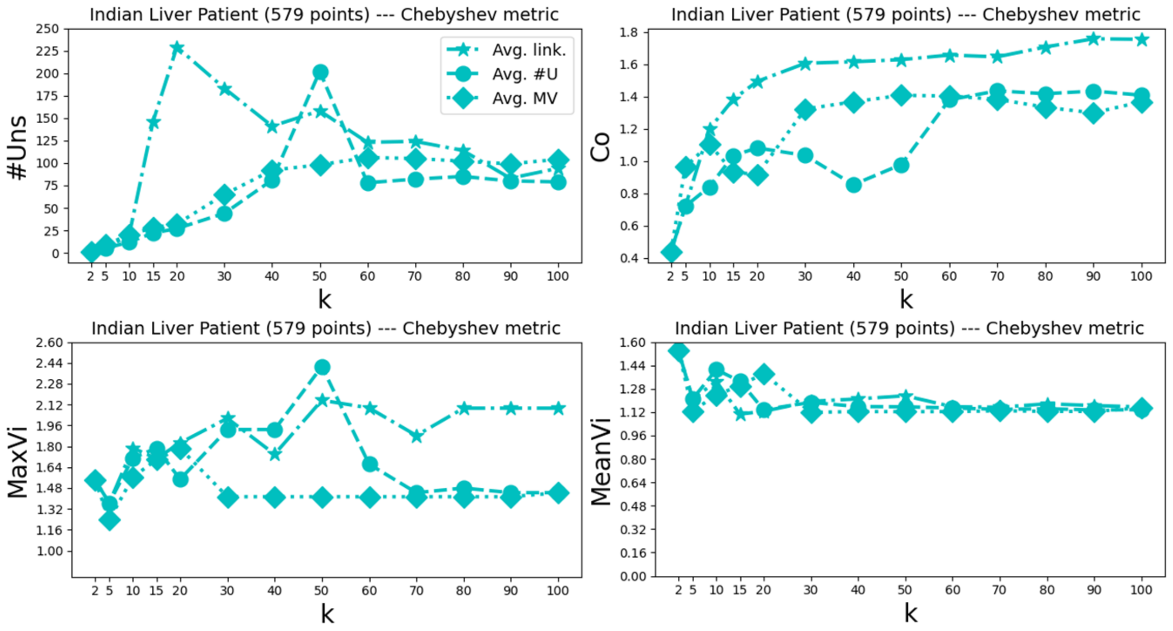

Figure 43. Indian Liver Patient data set with Chebyshev metric: the various measures for the clusterings produced by average linkage clustering and the two variants of our heuristic approach.

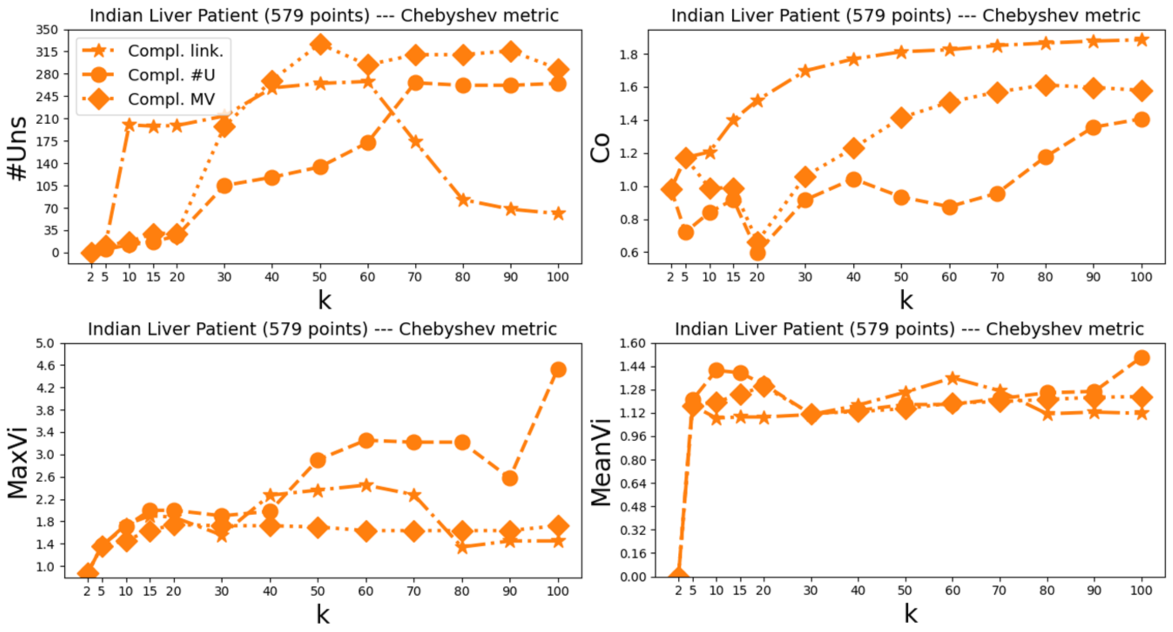

Figure 44. Indian Liver Patient data set with Chebyshev metric: the various measures for the clusterings produced by complete linkage clustering and the two variants of our heuristic approach.

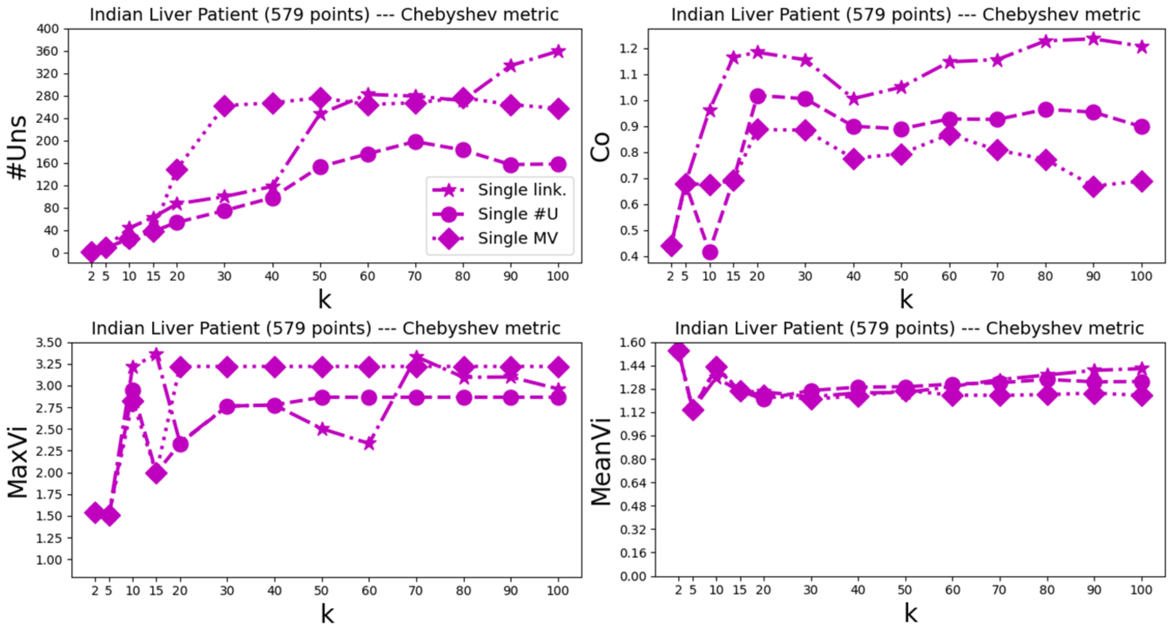

Figure 45. Indian Liver Patient data set with Chebyshev metric: the various measures for the clusterings produced by single linkage clustering and the two variants of our heuristic approach.

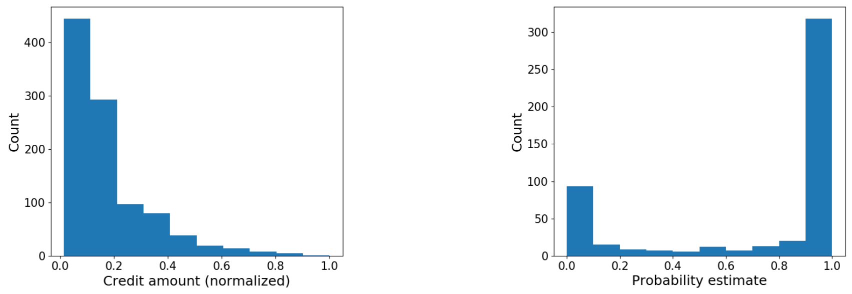

Figure 46. Histograms of the data sets used in the experiments of Appendix H. **Left:** The credit amount (one of the 20 features in the German credit data set; normalized to be in $[0, 1]$) for the 1000 records in the German credit data set. Note that there are only 921 unique values. **Right:** The estimated probability of having a good credit risk for the second 500 records in the German credit data set. The estimates are obtained from a multi-layer perceptron trained on the first 500 records in the German credit data set.

Table 1. Experiment on German credit data set. Clustering 1000 people according to their credit amount. Target cluster sizes $t_i = \frac{1000}{k}$, $i \in [k]$. NAIVE=naive clustering that matches the target cluster sizes, DP=dynamic programming approach of Appendix F, k-MEANS= k-means initialized with medians of the clusters of the naive clustering, k-ME++= k-means++. Results for k-ME++ averaged over 100 runs. Best values in bold.

		# UNS	MAXVI	MEANVI	OBJ	CO
$k = 5$	NAIVE	105	2.95	1.47	**0**	**0.24**
	DP	**0**	**1.0**	**0**	172	0.28
	k-MEANS	1	**1.0**	1.0	170	0.28
	k-ME++	1.11	1.01	0.61	275.3	0.29
$k = 10$	NAIVE	113	2.16	1.3	**0**	**0.28**
	DP	**0**	**1.0**	**0**	131	0.3
	k-MEANS	4	1.01	1.01	136	0.29
	k-ME++	3.16	1.02	0.99	159.47	0.29
$k = 20$	NAIVE	92	3.17	1.3	**0**	0.3
	DP	**0**	**1.0**	**0**	37	0.3
	k-MEANS	5	1.01	1.01	37	0.29
	k-ME++	7.44	1.06	1.03	105.06	**0.26**
$k = 50$	NAIVE	101	2.6	1.3	**0**	0.32
	DP	**0**	**1.0**	**0**	8	0.3
	k-MEANS	18	1.26	1.06	10	0.3
	k-ME++	12.7	1.19	1.05	50.22	**0.26**

H. Experiments on 1-dimensional Euclidean data sets

In this section we present experiments with our dynamic programming (DP) approach of Appendix F for 1-dim Euclidean data sets. We used the German Credit data set (Dua and Graff, 2019). It comprises 1000 records (corresponding to human beings) and for each record one binary label (good vs. bad credit risk) and 20 features.

In our first experiment, we clustered the 1000 people according to their credit amount, which is one of the 20 features. A histogram of the data can be seen in the left plot of Figure 46. We were aiming for k-clusterings with clusters of equal size (i.e., target cluster sizes $t_i = \frac{1000}{k}$, $i \in [k]$) and compared our DP approach with $p = \infty$ to k-means clustering as well as a naive clustering that simply puts the t_1 smallest points in the first cluster, the next t_2 many points in the second cluster, and so on. We considered two initialization strategies for k-means: we either used the medians of the clusters of the naive clustering for initialization (thus, hopefully, biasing k-means towards the target cluster sizes) or we ran k-means++ (Arthur and Vassilvitskii, 2007). For the latter we report average results obtained from running the experiment for 100 times. In addition to the four quantities # Uns, MaxVi, MeanVi and Co considered in the experiments of Section 6.1 / Appendix G.4 - G.6 (defined at the beginning of Section 6), we report Obj ("objective"), which is the value of the objective function of (3) for $p = \infty$. Note that k-means / k-means++ yields contiguous clusters and Obj is meaningful for all four clustering methods that we consider.

The results are provided in Table 1, where we consider $k = 5$, $k = 10$, $k = 20$ or $k = 50$. As expected, for the naive clustering we always have Obj $= 0$, and for our DP approach (DP) we have # Uns $= 0$, MaxVi ≤ 1 and MeanVi $= 0$. Most interesting to see is that both versions of k-means yield almost perfectly stable clusterings when k is small and moderately stable clusterings when $k = 50$ (with k-means++ outperforming k-means).

In our second experiment, we used the first 500 records of the data set to train a multi-layer perceptron (MLP) for predicting the label (good vs. bad credit risk). We then applied the MLP to estimate the probabilities of having a good credit risk for the other 500 people. A histogram of the probability estimates is shown in the right plot of Figure 46. We used the same clustering methods as in the first experiment to cluster the 500 people according to their probability estimate. We believe that such a clustering problem may arise frequently in practice (e.g., when a bank determines its lending policy) and that IP-stability is highly desirable in this context.

Table 2 and Table 3 provide the results. In Table 2, we consider uniform target cluster sizes $t_i = \frac{500}{k}$, $i \in [k]$, while in Table 3 we consider various non-uniform target cluster sizes. The interpretation of the results is similar as for the first experiment of Section 6.2. Most notably, k-means can be quite unstable with up to 33 data points being unstable when k is large, whereas k-means++ produces very stable clusterings with not more than three data points being unstable. However, k-means++ performs very poorly in terms of Obj, which can be almost ten times as large as for k-means and our dynamic programming approach (cf. Table 2, $k = 50$).

The MLP that we used for predicting the label (good vs. bad credit risk) in the second experiment of Section 6.2 has three hidden layers of size 100, 50 and 20, respectively, and a test accuracy of 0.724.

Table 2. Experiment on German credit data set. Clustering the second 500 people according to their estimated probability of having a good credit risk. Target cluster sizes $t_i = \frac{500}{k}$, $i \in [k]$. NAIVE=naive clustering that matches the target cluster sizes, DP=dynamic programming approach of Appendix F, k-MEANS= k-means initialized with medians of the clusters of the naive clustering, k-ME++= k-means++. Results for k-ME++ averaged over 100 runs. Best values in bold.

TARGET CLUSTER SIZES			# UNS	MAXVI	MEANVI	OBJ	CO
$k = 5$	$t_1 = \ldots = t_5 = 100$	NAIVE	197	58.28	9.52	**0**	0.34
		DP	**0**	**0.99**	**0**	214	**0.29**
		k-MEANS	1	1.02	1.02	212	**0.29**
		k-ME++	0.67	1.01	0.62	220.72	0.3
$k = 10$	$t_1 = \ldots = t_{10} = 50$	NAIVE	162	10.27	3.06	**0**	0.34
		DP	**0**	**0.98**	**0**	217	**0.29**
		k-MEANS	8	1.25	1.09	207	0.34
		k-ME++	1.11	1.02	0.87	248.86	**0.29**
$k = 20$	$t_1 = \ldots = t_{20} = 25$	NAIVE	116	9.64	2.09	**0**	0.34
		DP	**0**	**1.0**	**0**	155	0.29
		k-MEANS	33	2.13	1.38	95	0.3
		k-ME++	2.53	1.07	0.97	239.55	**0.28**
$k = 50$	$t_1 = \ldots = t_{50} = 10$	NAIVE	73	3.8	1.78	**0**	0.34
		DP	**0**	**1.0**	**0**	24	0.3
		k-MEANS	28	2.29	1.25	13	0.3
		k-ME++	3.56	1.24	1.08	233.09	**0.23**

Table 3. Experiment on German credit data set. Clustering the second 500 people according to their estimated probability of having a good credit risk. Various non-uniform target cluster sizes. NAIVE=naive clustering that matches the target cluster sizes, DP=dynamic programming approach of Appendix F, k-MEANS= k-means initialized with medians of the clusters of the naive clustering, k-ME++= k-means++. Results for k-ME++ averaged over 100 runs. Best values in bold.

TARGET CLUSTER SIZES			# UNS	MAXVI	MEANVI	OBJ	CO
$k = 12$	$t_i = \begin{cases} 50 & \text{for } 3 \leq i \leq 10 \\ 25 & \text{else} \end{cases}$	NAIVE	188	12.85	3.22	**0**	0.34
		DP	**0**	**0.97**	**0**	232	**0.29**
		k-MEANS	3	1.05	1.04	217	**0.29**
		k-ME++	1.44	1.03	0.77	254.81	**0.29**
$k = 12$	$t_1 = t_{12} = 10$, $t_2 = t_{11} = 15$, $t_3 = t_{10} = 25$, $t_4 = t_9 = 50$, $t_5 = t_8 = 50$, $t_6 = t_7 = 100$	NAIVE	251	65.99	6.56	**0**	0.33
		DP	**0**	**0.97**	**0**	247	**0.29**
		k-MEANS	14	1.52	1.16	238	0.3
		k-ME++	1.43	1.03	0.83	270.09	**0.29**
$k = 20$	$t_i = \begin{cases} 10 & \text{for } i = 1, 3, 5, \ldots \\ 40 & \text{for } i = 2, 4, 6, \ldots \end{cases}$	NAIVE	189	137.31	6.64	**0**	0.35
		DP	**0**	**1.0**	**0**	140	0.29
		k-MEANS	30	1.91	1.28	91	0.3
		k-ME++	2.62	1.07	0.98	224.26	**0.28**
$k = 20$	$t_i = \begin{cases} 115 & \text{for } i = 10, 11 \\ 15 & \text{else} \end{cases}$	NAIVE	224	215.88	13.01	**0**	0.34
		DP	**0**	**1.0**	**0**	165	0.29
		k-MEANS	51	3.04	1.67	112	0.3
		k-ME++	2.44	1.07	0.99	249.79	**0.28**