Scalable Algorithms for Individual Preference Stable Clustering

Ron Mosenzon
TTIC
ron.mosenzon@ttic.edu

Ali Vakilian
TTIC
vakilian@ttic.edu

Abstract

In this paper, we study the individual preference (IP) stability, which is an notion capturing individual fairness and stability in clustering. Within this setting, a clustering is α-IP stable when each data point's average distance to its cluster is no more than α times its average distance to any other cluster. In this paper, we study the natural local search algorithm for IP stable clustering. Our analysis confirms a $O(\log n)$-IP stability guarantee for this algorithm, where n denotes the number of points in the input. Furthermore, by refining the local search approach, we show it runs in an almost linear time, $\tilde{O}(nk)$.

1 INTRODUCTION

In numerous human-centric clustering applications, the data points being clustered often represent agents each having distinct preferences. In these contexts, ensuring fairness and stability necessitates clustering approaches that prioritize individual satisfaction of each agent over optimizing a global objective. One such notion is that of *individual preference stability (IP stability)*, which was recently introduced by Ahmadi et al. (2022). A dataset's clustering is IP stable when each data point finds its own cluster's average distance to be smaller than the average distance to any other cluster. This essentially means that every individual has a preference for its designated cluster, ensuring stability. The notion is closely related to envy-free clustering. More formally, given a metric space (X, d) on n points and a partition of X into k disjoint non-empty clusters $(C_1, \ldots, C_k)$, we say that: (1) a point $p \in C_i$ is α-envious of a cluster C_j if p's average distance to the points in its own cluster (i.e., C_i) is more than α times larger than its average distance to the points in C_j; and, (2) the partition is an α-IP stable clustering if no point is α-envious of any cluster.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

Perhaps one of the most natural way to find an α-IP stable clustering is the following local search procedure: Start with an arbitrary k-clustering of X, and, as long as there exists some point p that α-envies some cluster C, choose such a pair (p, C) and swap the point p from its current cluster to the cluster C. The study of local search for IP stable clustering is conceptually interesting for several reasons. Firstly, its simplicity suggests a high likelihood of practical effectiveness. Secondly, it offers insights into the natural emergence of IP stable clusterings in real-world scenarios. For instance, in situations where agents might gravitate towards another cluster with more similar members, the dynamics resemble the local search approach, thereby leading to the formation of an IP stable clustering. An example of this dynamic can be seen in social networks, where individuals often cluster with those who resonate with similar viewpoints or passions.

In this paper, we analyze the natural local search algorithm for this problem (Algorithm 1), and bound its approximation guarantee for general metric.

Algorithm 1: Local Search Meta-Algorithm.

```
   Data: $(X, d)$, $k \le n = |X|$, and $\alpha \ge 1$.
   Result: an $\alpha$-IP stable $k$-clustering of $X$.
1  initialize an arbitrary $k$-clustering $\mathcal{C}$ of $X$.
2  while exists $p$ and $C'$ s.t.
     $\mathrm{avg}(p, C(p) \setminus \{p\}) > \alpha \cdot \mathrm{avg}(p, C')$ do
3      move $p$ from $C = C(p)$ to the cluster $C'$
4  end
5  return the current clustering
```

1.1 Problem Definition and Background

Given a metric space (X, d) with $|X| = n$, a k-clustering of (X, d) is a partition of X into k disjoint non-empty clusters. Furthermore, for any non-empty subset $S \subseteq X$ and point $p \in X$, we use $\text{avg}(p, S)$ to denote the average distance of p to the points in S,

$$\text{avg}(p, S) \overset{\text{def}}{=} \frac{1}{|S|} \sum_{p' \in S} d(p, p').$$

Definition 1 (IP Stability (Ahmadi et al., 2022))**.** Given a metric space (X, d) with $|X| = n$. A clustering $\mathcal{C}$ of X is said to be α-IP stable if for every point $p \in X$ and every cluster $C' \in \mathcal{C}$ where $C' \neq C(p)$, either $C(p) = \{p\}$ or $\text{avg}(p, C(p) \setminus \{p\}) \leq \alpha \cdot \text{avg}(p, C')$.

Ahmadi et al. (2022) showed that, given a metric space (X, d) and a desired number of clusters k, it is not always possible to find a 1-IP stable k-clustering of X, as such a clustering may not exists. Furthermore, it is NP-hard to decide whether a 1-IP clustering exists for a given (X, d) and k. This NP-hardness has motivated the study of α-IP stability for $\alpha > 1$. Ahmadi et al. (2022) proposed an $O(n)$-IP stable clustering for general metrics, by applying the standard $O(n)$-distortion embedding to one-dimensional Euclidean space (Matoušek, 1990, Theorem 2.1). They also gave a bicriteria approximation that discards an ε-fraction of the input and outputs a $O\left(\frac{\log^2 n}{\varepsilon}\right)$-IP stable clustering for the remaining. Very recently, Aamand et al. (2023), via a clever ball carving technique, designed an algorithm that outputs an $O(1)$-IP stable clustering in all metric instance.

1.2 Our Results

Our main contribution is the analysis of the natural and practical local search algorithm for IP-stable clustering.

Simple Local Search. More precisely, we show that there exists $\alpha = O(\log n)$ such that the natural local search meta-algorithm (Algorithm 1) always terminates when used with this α.

Theorem 1. *For every $n \geq 3$, the natural local search algorithm (Algorithm 1) always terminates with parameter $\alpha \geq 2 \log n$ on any metric space with n points, regardless of the value of k.*

Even though Theorem 1 guarantees that the local search algorithm successfully finds an $O(\log n)$-IP stable clustering, it does not guarantee polynomial time running time. In fact, the worst-case runtime of this simple local search is $n^{\Theta(k)}$. This motivates our second main result, which is a fast algorithm that outputs a $O(\log n)$-IP stable k-clustering of any metric space.

Fast Algorithm. Building on the standard local search algorithm, we construct an $\tilde{O}(nk)$-time algorithm that finds a $O(\log n)$-IP stable k-clustering. Specifically, in the practical settings where $k = \text{polylog}(n)$, the running time of this algorithm is near-linear in the number of points. We remark that the running time is sub-linear in the input size, since the input consists of the pairwise distances of points. Our algorithm improves upon the runtime of (Aamand et al., 2023) by a factor of n.

Theorem 2. *There exists a Monte-Carlo randomized algorithm that, given a metric space (X, d) with $|X| = n$ and a desired number of clusters $k \leq n$, finds a $O(\log n)$-IP stable k-clustering of (X, d) in time $\tilde{O}(nk)$. The error probability can be made to be an arbitrarily small polynomial at the cost of only a* polylog(n) *blowup in the running time.*

Additionally, if the points in X are located in small number $m \ll n$ of distinct *locations*, then the algorithm from Theorem 2 can be easily modified so that the running time will scale with m rather than with n. (i.e. the running time becomes $\tilde{O}(mk)$, but the $\tilde{O}$ notation still hides polylog(n) factors.) This modification is performed by treating each location as if it is one "weighted" point, which means that the modified algorithm also has the "consistency" property that points in the same location are guaranteed to belong to the same cluster of the output k-clustering. We remark that unlike our result, the runtime of the algorithm of Aamand et al. (2023) scale with the total weight of points in the input and fails to satisfy the natural consistency property.

Bounding Approximation Factor. While the described results so far, as well as the guarantees of prior works, only provide an absolute IP-stability bound, Aamand et al. (2023) posed an open question whether it is possible to return a clustering whose IP-stability guarantee is within a constant factor of the the best achievable IP-stability guarantee for the given instance. In particular, if a given metric space (X, d) admits an α^*-IP stable clustering with $\alpha^* = o(1)$, it is possible to find a $O(\alpha^*)$-IP stable clustering of the instance. The problem is closely related to a notion of stability for clustering studied in a line of work by (Daniely et al., 2012; Balcan et al., 2013). However, their guarantee requires a lower bound of βn on the size of the minimum cluster in the underlying solution and the runtime of their algorithm exponentially depends on $\frac{n}{\beta} > k$. Here, we answer this question too.

Theorem 3. *There exists a deterministic algorithm that, given a metric space (X, d) with $|X| = n$ and a desired number of clusters k, runs in polynomial time and returns a k-clustering $\mathcal{C}$ with the guarantee that, if there exists a α^*-IP stable k-clustering of X for $\alpha^* <$* 0.001*, then $\mathcal{C}$ is $(3\alpha^*)$-IP stable.*

In particular, together with the $O(1)$-IP stable guarantee of the recent work of Aamand et al. (2023), the above theorem implies $O(1)$-approximate IP stable clustering; there exists an algorithm that on every metric instance (X, d) returns an $O(\alpha^*)$-IP stable clustering where α^* is the smallest value for which (X, d) admits a α^*-IP stable clustering (note that α^* can be any value, either ≥ 1 or < 1).

Other IP Stability Functions. Moreover, we study the IP-stability with respect to functions other than average. Aamand et al. (2023) proposed the general notion of (α, f)-IP stability (or f-IP stability for brevity) as follows.

Definition 2. Given a metric space (X, d) and a function $f : X \times 2^X \to \mathbb{R}_{\geq 0}$, a clustering $\mathcal{C}$ of (X, d) is (α, f)-IP stable if, for every point $p \in X$, either $C(p) = \{p\}$ or for every cluster $C' \neq C(p)$ in $\mathcal{C}$,

$$f(p, C(p)) \leq \alpha \cdot f(p, C').$$

We show that our analysis of the natural local search for avg-IP stability, and our fast adaptation of the standard local search, can be extended to other special cases of f-IP stability through careful modification. We use this framework in order to prove that the natural local search for median-IP stability outputs an $O(1)$-approximate median-IP stable clustering, and can be modified to run in polynomial time.

Theorem 4. *For every metric space (X, d), every non-empty subset $S \subseteq X$, and every point $p \in X$, let* $\mathrm{median}(p, S)$ *denote the $\lceil |S|/2 \rceil$th smallest of the $|S|$ values $\{d(p, p')\}_{p' \in S}$. Then, there exists a deterministic algorithm running in polynomial time that for a desired number of clusters $2 \leq k \leq |X|$, computes a $(O(1), \mathrm{median})$-IP stable k-clustering of X.*

Furthermore, we analyze max-IP stability, which is the case where $f(p, C) \overset{\text{def}}{=} \max_{p' \in C} d(p, p')$, and show that every metric space admits an exact max-IP stable k-clustering for every value of k.

Theorem 5. *For every metric space (X, d) and every $2 \leq k \leq |X|$, there exists a $(1, \max)$-IP stable k-clustering of X.*

This in particular, improves upon the 3-approximate guarantee of Aamand et al. (2023) for max-IP stable k-clustering.

Additionally, we note that for a related problem of min-IP stable k-clustering, exact polynomial time algorithms are known (Aamand et al., 2023; Laber and Murtinho, 2023).

1.3 Our Techniques

To prove our main result (Theorem 1), we employ a potential function argument. Specifically, we show that if there exists a potential function $\Phi : 2^X \to \mathbb{R}_{\geq 0}$ with the property that, for every $S \subset X$ and every $p \in X \backslash S$,

$$\mathrm{avg}(p, S) \leq \Phi(S \cup \{p\}) - \Phi(S) \leq \alpha \cdot \mathrm{avg}(p, S), \quad (1)$$

then every step of the natural local search, Algorithm 1, reduces the sum of potentials of the clusters in the maintained k-clustering. Furthermore, we show that when $\alpha = O(\log n)$, such a potential function exists. The potential function $\Phi(C)$ that we choose is defined in terms of the average distances of the points $p \in C$ to the cluster C; $\sum_{p \in C} \mathrm{avg}(p, C)$. A key observation in the proof of Equation (1) is Lemma 1 which relates the value of $\mathrm{avg}(p, C)$ to the value of $\Phi(C)$.

Faster Algorithm. At its core, the algorithm from Theorem 2 is based on the natural local search, but with some modifications that speed it up significantly. The idea behind the main adaptation is to identify the cases in which a local update does not make large enough progress with respect to the potential function Φ, and use a different type of local step to handle these cases. This modification by itself reduces the running time to $\tilde{O}(n^2 k)$. Shaving off the last factor of n requires a few more modifications. Essentially, we exploit the fact that $\mathrm{avg}(p, C)$ values do not often change drastically, enabling the use of calculated values for $\mathrm{avg}(p, C)$ as estimates in multiple forthcoming steps, instead of refreshing these values after every single step. Furthermore, once these previously computed $\mathrm{avg}(p, C)$ no longer provide good enough estimates for the current values, we show how to quickly recompute these values, employing a statistical technique known as Importance Sampling.

Approximate IP Stability In contrast to our other results, the algorithm for Theorem 3 is not a local search algorithm. Instead, it uses dynamic programming, while exploiting the structure that a highly IP-stable clustering imposes on the metric space. Indeed, there is little hope of using a local search algorithm similar to Algorithm 1 with $\alpha < 1$, since it may get stuck repeatedly moving the same point p between two clusters that are equidistant from p. We remark that, with a more optimized implementation of the algorithm from Theorem 3, a running time of $O(n^2)$ is achievable.

Median-IP Stability The median-IP stable algorithm of Theorem 4 is a simplified version of the algorithm from Theorem 2. In fact, in the appendix, we present a framework for constructing such algorithms for general IP stability functions, provided that an appropriate potential function exists. While Theorem 4 only guarantees polynomial running time, we note that an upper-bound of $\tilde{O}(n^3)$ can be shown with a tighter analysis of the same algorithm, and that an algorithm with a running time of $\tilde{O}(n^2k)$ can be achieved if the stability guarantee is relaxed from $\alpha = O(1)$ to $\alpha = O(\log n)$.

Max-IP Stability The result presented in Theorem 5 is derived via a natural local search algorithm. Analogously to Algorithm 1, there is no guarantee that this local search algorithm runs in polynomial time.

1.4 Related Work

Clustering Stability. Designing efficient clustering algorithms that consider various notions of stability is a well-studied problem. Notably, the concept of "average stability" (Balcan et al., 2008) is particularly relevant to our model. For a comprehensive survey on stability in clustering, refer to (Awasthi and Balcan, 2014). While existing research on clustering stability (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) mainly drives the development of faster algorithms by leveraging stability conditions, IP stable clustering explores whether such stability properties can be identified in generic instances, either exactly or approximately.

Individual Fairness in Clustering. As introduced by (Ahmadi et al., 2022), a primary motivation for IP-stability lies in its connection to individual fairness. Individual fairness was initially formulated in the seminal work of Dwork et al. (2012), aiming to ensure classification problems provided "similar predictions" for "similar points". In recent years, several notions of individual fairness for clustering have been proposed (Jung et al., 2020; Anderson et al., 2020; Brubach et al., 2020; Chakrabarti et al., 2022; Kar et al., 2023). In a line of research by Jung et al. (2020); Mahabadi and Vakilian (2020); Negahbani and Chakrabarty (2021); Vakilian and Yalçıner (2022), a notion of fairness has been studied, wherein each point expects to find its closest center at a distance comparable to that from its (n/k)-th closest neighbor. Brubach et al. (2020, 2021) explored settings where individuals benefit from being clustered together, aiming to discover a clustering where all individuals gain equitable community benefits. Lastly, Anderson et al. (2020) introduced a formulation whereby the output is a distribution over centers, and "similar" points must have "similar" centers distributions.

Besides, clustering has also been explored in the context of various other fairness notions, including equitable representation in clusters (Chierichetti et al., 2017; Bera et al., 2019; Bercea et al., 2019; Backurs et al., 2019; Schmidt et al., 2019; Ahmadian et al., 2019; Dai et al., 2022), equitable representation in centers (Krishnaswamy et al., 2011; Chen et al., 2016; Krishnaswamy et al., 2018; Kleindessner et al., 2019; Chiplunkar et al., 2020; Jones et al., 2020; Hotegni et al., 2023), proportional fairness (Chen et al., 2019; Micha and Shah, 2020), and min-max fairness (Abbasi et al., 2021; Ghadiri et al., 2021; Makarychev and Vakilian, 2021; Chlamtáč et al., 2022; Ghadiri et al., 2022).

IP-stability is also relevant to the notion of *envy-freeness*. Within conventional fair division problems, such as cake cutting (Robertson and Webb, 1998; Procaccia, 2013) and rent division (Edward Su, 1999; Gal et al., 2016), envy-freeness requires that each participant should (weakly) favor their own allocation over that of others (Foley, 1966; Varian, 1973). More recently, the concept was studied for the problem of classification (Zafar et al., 2017; Balcan et al., 2019).

Hedonic Games in Clustering. Hedonic games represent another game-theoretic approach to studying clustering. This approach has been explored in several studies, including those by (Dreze and Greenberg, 1980; Bogomolnaia and Jackson, 2002; Elkind et al., 2020; Stoica and Papadimitriou, 2018). Within these frameworks, players strategically form coalitions, aiming to optimize their respective utilities. Our work diverges from this paradigm as we do not treat data points as selfish players.

1.5 Paper Outline

In Section 2, we analyze the correctness of the natural local search (Theorem 1). In Section 3, we present a slower but simpler version of the algorithm from Theorem 2, achieving a running time of $\tilde{O}(n^2k)$. This simpler variant highlights the most crucial modification that distinguishes the final $\tilde{O}(nk)$ time algorithm from the natural local search algorithm, without being obscured by the numerous details inherent in the full algorithm. In the appendix, we show how to shave off the last factor n from the running time, resulting in the fast algorithm from Theorem 2. All missing proofs are also in the appendix.

1.6 Preliminaries

In all algorithms in this paper, the input metric space (X, d) is given in the form of query access to the distance metric d, where each query takes $\Theta(1)$ time. We always use $n = |X|$ to denote the number of points in the input.

Definition 3 (average distance)**.** Given a metric space (X, d), a non-empty subset $S \subseteq X$, and a point $p \in X$, we let $\operatorname{avg}(p, S)$ denote the average distance of p to the points in S, $\operatorname{avg}(p, S) \overset{\text{def}}{=} \frac{1}{|S|} \sum_{p' \in S} d(p, p')$.

The following lemma is a crucial component in our analysis of the natural local search algorithm and the subsequent fast modifications of it. This lemma is derived in an elementary manner, utilizing the triangle inequality. We defer the proof to the appendix, where we prove a more general statement.

Lemma 1. *In every metric space (X, d), for every non-empty subset $S \subseteq X$, and every point $p \in X$,*

$$\frac{1}{|S|} \sum_{p' \in S} \operatorname{avg}(p', S) \leq 2 \cdot \operatorname{avg}(p, S)$$

2 ANALYSIS OF THE NATURAL LOCAL SEARCH

The goal of this section is to prove Theorem 1. The main technique is a potential function argument. More precisely, given a metric space (X, d) and a parameter k, we define a potential function on the set of all possible k-clusterings. Next, we show that each step of the natural local search algorithm reduces the potential of the clustering maintained by the local search. To achieve this, we define a potential function on clusters $\Phi : 2^X \to \mathbb{R}_{\geq 0}$, and say that the potential of a clustering $\mathcal{C}$ is the sum of the potentials of its clusters $\Phi(\mathcal{C}) \overset{\text{def}}{=} \sum_{C \in \mathcal{C}} \Phi(C)$. Our main important observation is the following:

Observation 1. *Given $\alpha \geq 1$ and a metric space (X, d), if a potential function $\Phi : (2^X \setminus \{\emptyset\}) \to \mathbb{R}_{\geq 0}$ satisfies that, for every non-empty set $S \subset X$ and every point $p \notin S$,*

$$\operatorname{avg}(p, S) \leq \Phi(S \cup \{p\}) - \Phi(S) \leq \alpha \cdot \operatorname{avg}(p, S),$$

then each step of the natural local search algorithm with parameter α reduces the value of $\sum_{C \in \mathcal{C}} \Phi(C)$, where $\mathcal{C}$ is the clustering maintained by the algorithm.

The main challenge in the proof of Theorem 1 is to show the existence of a potential function satisfying the required property of Observation 1 with $\alpha = 2 \log n$.

Proof of Observation 1. The property of the potential function implies that removing a point p from a cluster $C \neq \{p\}$ decreases the potential of that cluster by at least $\operatorname{avg}(p, C \setminus \{p\})$. This property also implies that adding a point p to a cluster C' increases its potential by at most $\alpha \cdot \operatorname{avg}(p, C')$. Note that in each step of the local search algorithm a point p is removed from a cluster $C \neq \{p\}$ and is added to a cluster $C' \neq C$ such that $\operatorname{avg}(p, C \setminus \{p\}) > \alpha \cdot \operatorname{avg}(p, C')$. Hence, in each step, the decrease in the potential of the respective cluster C is larger than the increase in the potential of the respective cluster C', and there is no change in the potential of any other clusters. This implies that each local update reduces the total potential of the clustering, as desired. □

2.1 Potential Function

In this subsection, we prove that there exists a potential function $\Phi : (2^X \setminus \{\emptyset\}) \to \mathbb{R}_{\geq 0}$ that satisfies the property from Observation 1 for every $\alpha \geq 2 \log n$.

Theorem 6. *For every metric space (X, d) with $|X| = n \geq 3$, there exists a potential function $\Phi : (2^X \setminus \{\emptyset\}) \to \mathbb{R}_{\geq 0}$ on subsets of X with the property that, for every non-empty set $S \subset X$ and every point $p \notin S$,*

$$\operatorname{avg}(p, S) \leq \Phi(S \cup \{p\}) - \Phi(S) \leq 2 \log n \cdot \operatorname{avg}(p, S).$$

In order to prove the theorem, we will use the following potential function:

$$\Phi(S) = \log |S| \cdot \sum_{p \in S} \operatorname{avg}(p, S) = \frac{\log |S|}{|S|} \sum_{p,q \in S} d(p, q).$$

We provide a brief outline of the proof for the existence of a potential function here and include the full proof in the appendix.

Proof sketch of Theorem 6. We provide upper and lower bounds for $\Phi(S \cup \{p\}) - \Phi(S)$, the amount by which the potential of the set S increases when a point p is added to it. Let us define the function $f(x, y) = \frac{x \log y}{y}$ and use the notation $x_S = \sum_{p_1, p_2 \in S} d(p_1, p_2)$ and $y_S = |S|$. Note that $\Phi(S) \equiv f(x_S, y_S)$. Thus, using this notation, the quantity that we need to analyze is the amount of change in the value of $f(x_S, y_S)$ as the point p is added to S. Our goal is to show that this change is at least $\operatorname{avg}(p, S)$ and is at most $2 \log n \cdot \operatorname{avg}(p, S)$. At a high-level, we estimate this change by approximating the function f linearly, utilizing its derivative at the point (x_S, y_S). Note that, as f is linear in its first argument, and the value of y_S only changes by 1, it provides a "good" estimate for the amount of change in $f(x_S, y_S)$.

Moreover, note that by adding a point p to a set S, the value of x_S increases by exactly

$$\begin{aligned} x_{S\cup\{p\}} - x_S &= \sum_{p_1,p_2\in S\cup\{p\}} d(p_1,p_2) - \sum_{p_1,p_2\in S} d(p_1,p_2) \\ &= d(p,p) + 2\sum_{p'\in S} d(p,p') \\ &= 2|S|\cdot \mathrm{avg}(p,S), \end{aligned}$$

and the value of y_S increases by exactly 1. So,

$$\begin{aligned} &\Phi(S\cup\{p\}) - \Phi(S) \\ &\quad \simeq 2|S|\cdot \mathrm{avg}(p,S)\cdot \frac{\partial f}{\partial x}(x_S,y_S) + \frac{\partial f}{\partial y}(x_S,y_S) \end{aligned}$$

It is straightforward to verify that $\frac{\partial f}{\partial x}(x_S,y_S) = \frac{\log y_S}{y_S}$ and $\frac{\partial f}{\partial y}(x_S,y_S) = \frac{\log e - \log y_s}{y_S^2}x_S$. So,

$$\begin{aligned} &\Phi(S\cup\{p\}) - \Phi(S) \\ &\quad \simeq 2|S|\cdot \mathrm{avg}(p,S)\cdot \frac{\log y_S}{y_S} + \frac{\log e - \log y_S}{y_S^2}x_S \\ &\quad = 2\cdot \mathrm{avg}(p,S)\cdot \log y_S + \frac{\log e - \log y_S}{y_S^2}x_S, \end{aligned}$$

Furthermore, using this notation, Lemma 1 exactly translates to $\frac{x_S}{y_S^2} \le 2\,\mathrm{avg}(p,S)$. Since $\frac{x_S}{y_S^2} \ge 0$, when $y_S \ge e$,

$$\begin{aligned} \Phi(S\cup\{p\}) - \Phi(S) &\ge 2\cdot \mathrm{avg}(p,S)\cdot \log e, \text{ and} \\ \Phi(S\cup\{p\}) - \Phi(S) &\le 2\cdot \mathrm{avg}(p,S)\cdot \log y_S \\ &\le 2\log n\cdot \mathrm{avg}(p,S). \end{aligned}$$

□

Now, we are ready to prove Theorem 1.

Proof of Theorem 1. Given a metric space (X,d), Theorem 6 implies a potential function Φ that satisfies the condition of Observation 1 for $\alpha = 2\log n$. Thus, Observation 1 implies that $\sum_{C\in\mathcal{C}}\Phi(C)$ decreases at each step of the local search (Algorithm 1) when $\alpha \ge 2\log n$. Therefore, the state of the clustering $\mathcal{C}$ can never repeat. So, since the number of possible k-clusterings of (X,d) is finite, the algorithm must terminate. □

3 ADAPTING LOCAL SEARCH TO RUN IN POLYNOMIAL TIME

In this section, we present a simplified version of the algorithm from Theorem 2, with run-time $\tilde{O}(n^2k)$. This version presents most of the main ideas in the algorithm, and is much simpler. In the appendix, we present the fast algorithm, thus proving Theorem 2.

We first show why the natural local search is slow, and then present an overview of our approach to fix it, achieving the $\tilde{O}(n^2k)$-time algorithm.

While Observation 1 showed that by setting $\alpha = 2\log n$, after each local search step, moving $p\in C$ to a new cluster C', the total potential of the clustering decreases, it is not hard to see that the same analysis implies that if we instead set $\alpha = \gamma\cdot(2\log n)$, after each local search step, the potential decreases by at least $\left(1-\frac{1}{\gamma}\right)\cdot \mathrm{avg}(p, C\setminus\{p\})$.

However, the natural local search may still be notably slow since the value of $\mathrm{avg}(p, C\setminus\{p\})$ might be significantly smaller than the total potential of the clustering, denoted by $\Phi(\mathcal{C}) = \sum_{C\in\mathcal{C}}\Phi(C)$. Moreover, there exist instances where, for all $p\in X$ that envy any cluster, $\mathrm{avg}(p, C(p))$ is small. This scenario occurs when a large fraction of $\Phi(\mathcal{C})$ is due to $\mathrm{avg}(p', C(p'))$ for points p' that do not envy any other cluster.

In this section we remedy this issue of the natural local search, thus achieving an $\tilde{O}(n^2k)$-time algorithm. To do this, we show that when the value of $\mathrm{avg}(p, C\setminus\{p\})$ in the current step is significantly smaller than $\Phi(\mathcal{C})$, then there is a different "modififcation" of the current clustering that significantly reduces the value of $\Phi(\mathcal{C})$. A key observation is that any cluster C^* can efficiently be split into two clusters C_1^* and C_2^* such that $\Phi(C_1^*)+\Phi(C_2^*)$ is smaller than $\Phi(C^*)$. Indeed, by splitting a cluster C^* into two equally-sized randomly chosen parts,

$$\begin{aligned} &\mathbb{E}\left[\Phi(C_1^*)+\Phi(C_2^*)\right] \\ =&\frac{\log(|C^*|/2)}{|C^*|/2}\cdot \sum_{p,p'\in C^*} d(p,p')\cdot \Pr[p,p'\in C_1^* \vee p,p'\in C_2^*] \\ \le&\frac{\log(|C^*|/2)}{|C^*|/2}\cdot \sum_{p,p'\in C^*} d(p,p')\cdot \frac{1}{2} \\ =&\frac{\log(|C^*|)-1}{|C^*|}\cdot \sum_{p,p'\in C^*} d(p,p') \\ =&(1-\frac{1}{\log|C^*|})\cdot \Phi(C^*). \end{aligned}$$

Intuitively speaking, if we randomly split a cluster C^* such that $\Phi(C^*) \ge \frac{\Phi(\mathcal{C})}{k}$, we can expect $\Phi(\mathcal{C})$ to decrease by a factor of approximately $\left(1-\frac{1}{k\log|C^*|}\right)$. However, splitting a cluster is not a feasible operation on its own since it increases the number of clusters. So, to leverage the splitting operation, we also consider a "merge", particularly beneficial if merging two existing clusters increases the total potential by less than the amount decreased by the split.

Indeed, we demonstrate that a "merge and split" step exists when the standard local search step scarcely decreases the potential function. When moving $p \in C$ to a new cluster C' only decreases the potential by a little, the point p is "close" to both clusters C and C'. This implies that C and C' are also "close to each other". Furthermore, we define a distance measure between two clusters providing an upper bound on the amount by which merging the clusters will increase the total potential $\Phi(\mathcal{C})$.

To summarize, the high-level idea is that if the current step of the natural local search only marginally reduces $\Phi(\mathcal{C})$, then both $\text{avg}(p, C(p) \setminus \{p\})$ and $\text{avg}(p, C')$ are "small". Consequently, we can merge the clusters $C(p)$ and C', incurring only a "small" increase in $\Phi(\mathcal{C})$. In such cases, $C(p)$ and C' can be merged and a cluster C^*, for which $\Phi(C^*) \geq \Phi(\mathcal{C})/k$, can instead be split. Then, we show that the increase in the total potential from the merge is offset by the potential's decrease due to the split operation.

The above ideas culminate in an algorithm that can quickly make progress on reducing the potential of its maintained clustering by repeatedly finding a point p that is envious, and performing either the "swap step" of the natural local search, or a "merge and split" step. The algorithm stops when no point is α-envious for $\alpha = O(\log n)$, which implies that the current clustering is α-IP stable. We show that the described algorithm reduces the potential of the clustering by a constant factor in time $\tilde{O}(n^2 k)$. One last issue is that the ratio between the potential of the initial clustering and the potential of the final clustering can be very large, resulting in a possibly $\Omega(\log(\frac{\Phi(\mathcal{C}_{\text{initial}})}{\Phi(\mathcal{C}_{\text{final}})}))$ iterations of "reducing the clustering's potential by a constant factor". To get around this issue, we show that initializing with an approximate k-center solution guarantees that $\Phi(\mathcal{C}_{\text{initial}}) \leq \text{poly}(n) \cdot \Phi(\mathcal{C}_{\text{final}})$; hence, the total number of iterations in the "modified" local search is $O(\log n)$.

The previous discussion shows that formally analyzing the algorithm requires the next three technical lemmas. In Section 3.1, we will present the algorithm and its analysis using the following three lemmas. We will prove these lemmas in the appendix. We remark that in the appendix, we show that the procedure from Lemma 2 runs in time $\tilde{O}(n)$.

Lemma 2. *There exists a $\tilde{O}(n^2)$-time Monte-Carlo randomized algorithm that, given a metric space (X, d) with n points and a $(k-1)$-clustering $\mathcal{C}$, finds a cluster $C^* \in \mathcal{C}$ and a partition (C_1^*, C_2^*) of C^* such that $\Phi(C_1^*) + \Phi(C_2^*) \leq \Phi(C^*) - \Omega(\frac{\Phi(\mathcal{C})}{k \log n})$. The error probability can be made as small as $O(\frac{1}{n^c})$, for any predefined constant c.*

Lemma 3. *Given a metric space (X, d) and two disjoint clusters $C, C' \subseteq X$, the increase in the potential Φ after merging C and C' is at most,*

$$\frac{2 \log n}{\max\{|C|, |C'|\}} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2). \tag{2}$$

Furthermore, for every point $p \in X$, Eq. (2) is at most

$$2 \log n \cdot \min\{|C|, |C'|\} \cdot (\text{avg}(p, C) + \text{avg}(p, C')).$$

Lemma 4. *Consider a metric space (X, d) with n points and a k-clustering $\mathcal{C}$. If $\mathcal{C}$ is a $O(1)$-approximate solution for k-center on X, then it* $\text{poly}(n)$*-approximately minimizes the potential function Φ; more precisely, $\Phi(\mathcal{C}) \leq \text{poly}(n) \cdot \Phi(\mathcal{C}^*)$, where $\mathcal{C}^*$ is a k-clustering minimizing Φ.*

Note that there exists a 2-approximation algorithm for k-center, e.g., (Gonzalez, 1985), that runs in time $O(nk)$. So, the initialization runs in $O(nk)$ time.

3.1 Algorithm and Analysis

We show that Algorithm 2 runs in time $\tilde{O}(n^2 k)$. This is a slower but simpler variant of the algorithm from Theorem 2.

In our analysis, we will proceed under the assumption that the split procedure of Lemma 2 does not encounter any errors. This assumption is justified since the error probability of the split procedure can be made to be an arbitrarily small polynomial at the cost of only a polylog(n) blowup in the running time.

In the analysis of this algorithm, we refer to each iteration of the while loop as a *step* of the algorithm. We refer to those iterations where the condition of the if statement in line 6 was met as *swap* steps, and to those iterations where this condition was not met as *merge and split* steps. It is easy to see that algorithm only terminates once it reaches an α-approximate IP stable clustering, so the entire analysis is focused on bounding the runtime by $\tilde{O}(n^2 k)$. We do this by proving the following two claims.

Claim 1. *Each swap step reduces the clustering's potential by a factor of $\left(1 - \tilde{\Omega}(\frac{1}{nk})\right)$, and each merge and split step reduces the potential by a factor of $\left(1 - \tilde{\Omega}(\frac{1}{k})\right)$.*

Claim 2. *Each swap step runs in time $\tilde{O}(n)$, and each merge and split step can be implemented in time $\tilde{O}(n^2)$. This includes the operations required to check the conditions of the while loop and the if statement.*

To see why these two claims imply $\tilde{O}(n^2 k)$ runtime of the algorithm, notice that, by Lemma 4, the total reduction in potential of the clustering across all steps is at most a poly(n) multiplicative factor. Thus,

Algorithm 2: Adapting the local search algorithm for a runtime of $\tilde{O}(n^2 k)$.

```
   Data: $(X,d)$, $n = |X|$, $k \le n$, $\alpha = 4\log n$
   Result: an $\alpha$-IP stable $k$-clustering of $(X,d)$
1  set $\mathcal{C}$ be the clustering returned by the greedy algorithm of $k$-center (Gonzalez, 1985) on $X$
2  for every $p \in X$ and $C \in \mathcal{C}$ do
3      compute $\operatorname{avg}(p, C)$
4  end
5  while exist $p \in X$ and $C' \in \mathcal{C}$ s.t. $C(p) \neq \{p\}$ and $\operatorname{avg}(p, C(p) \setminus \{p\}) > \alpha \cdot \operatorname{avg}(p, C')$ do
6      if $\operatorname{avg}(p, C(p) \setminus \{p\}) \ge \Omega(\frac{\Phi(\mathcal{C})}{k\log(n)})/(5n\log n)$ then
7          move $p$ from $C = C(p)$ to $C'$
8          for every $p' \in X$ do
9              update $\operatorname{avg}(p', C)$ and $\operatorname{avg}(p', C')$
10         end
11     else
12         $C'' \leftarrow C \cup C'$    ▷ merge clusters $C, C'$
13         $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C, C'\}) \cup \{C''\}$
14         Split($\mathcal{C}$)    ▷ a cluster $C^* \in \mathcal{C}$ is split into $C_1^*$ and $C_2^*$ by Split (as in Lemma 2)
15         $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C^*\}) \cup \{C_1^*, C_2^*\}$
16         for every $p' \in X$ and $C \in \{C'', C_1^*, C_2^*\}$ do
17             compute $\operatorname{avg}(p', C)$
18         end
19     end
20 end
21 return $\mathcal{C}$
```

Claim 1 implies that there are at most $\tilde{O}(nk)$ swap steps and at most $\tilde{O}(k)$ merge and split steps. Together with Claim 2, this implies that the algorithm spends at most $\tilde{O}(n^2k)$ time across all steps. Furthermore, the time that the algorithm spends outside of the while loop is at most $O(n^2k)$, since the greedy algorithm of k-center runs in time $O(nk)$ and calculating each $\operatorname{avg}(p, C)$ requires only $O(n)$ time.

Proof of Claim 1. Let $\mathcal{C}$ be the clustering at the beginning of a given step. We will now show that the total change in potential during this step is a decrease, by $\Phi(\mathcal{C}) \cdot \tilde{\Omega}(\frac{1}{nk})$ if the step is a swap step, and by $\Phi(\mathcal{C}) \cdot \tilde{\Omega}(\frac{1}{k})$ if the step is a merge and split step.

For a swap step, the same analysis from the proof of Observation 1 shows that removing the point p from the cluster C decreases the potential of C by at least $\operatorname{avg}(p, C \setminus \{p\})$, and that adding the point p to the cluster C' increases the potential of that cluster by at most $\alpha \cdot \operatorname{avg}(p, C')$, where $\alpha = 2\log n$ as in Theorem 6. However, unlike in Section 2, Algorithm 2 guarantees that $\operatorname{avg}(p, C \setminus \{p\}) > (4\log n) \cdot \operatorname{avg}(p, C')$, so the increase in the potential of the cluster C' is smaller than $\frac{1}{2}\operatorname{avg}(p, C \setminus \{p\})$. Thus, the total change in the potential of the clustering in this swap step is a decrease by more than $\frac{1}{2}\operatorname{avg}(p, C \setminus \{p\})$. Furthermore, since this is a swap step, the condition of the if statement in line 6 of the algorithm must have been met, which means that

$$\frac{1}{2}\operatorname{avg}(p, C \setminus \{p\}) \ge \Omega(\frac{\Phi(\mathcal{C})}{k\log(n)})/(10n\log n) = \Phi(\mathcal{C}) \cdot \tilde{\Omega}(\frac{1}{nk}).$$

Next, we analyze the merge and split step. First, note that splitting C^* decreases the potential by $\Omega(\frac{\Phi(\mathcal{C})}{k\log(n)})$. Second, Lemma 3 guarantees that merging C and C' increases the potential by at most

$$(2\log n) \cdot \min\{|C|, |C'|\} \cdot (\operatorname{avg}(p, C) + \operatorname{avg}(p, C')) \le (2\log n) \cdot n \cdot (\operatorname{avg}(p, C) + \operatorname{avg}(p, C')).$$

Since $\operatorname{avg}(p, C) \le \operatorname{avg}(p, C \setminus \{p\})$ and $\operatorname{avg}(p, C') < \operatorname{avg}(p, C \setminus \{p\})/(4\log n)$, this increase in the potential is at most $(\log n) \cdot n \cdot \frac{5}{2}\operatorname{avg}(p, C \setminus \{p\})$. Furthermore, since this is a merge and split step, the condition of the if statement in line 6 does not hold. Thus, if we set the value of $\Omega(\frac{\Phi(\mathcal{C})}{k\log(n)})$ in the if statement to be the same as the one guaranteed by Lemma 2, we get that the increase in the potential due to the merge is at most a half of the decrease in the potential due to the split. So, the overall change in the potential due to the merge and split step is a decrease by at least $\frac{1}{2} \cdot \Omega(\frac{\Phi(\mathcal{C})}{k\log(n)}) = \Phi(\mathcal{C}) \cdot \tilde{\Omega}(\frac{1}{k})$. □

Proof of Claim 2. We will start by explaining how to check the conditions of the while statement and the if statement in time $\tilde{O}(n)$. Then we analyze the running time of implementing lines 7-9 and lines 12-17. For the if statement, the only non-trivial operation involves calculating $\Phi(\mathcal{C})$. However, this can be efficiently computed using the equality $\Phi(\mathcal{C}) = \sum_{p' \in X} \log(|C(p')|) \cdot \operatorname{avg}(p', C(p'))$ and considering that the algorithm already maintains the values of $\operatorname{avg}(p', C(p'))$. The most challenging part is implementing the condition of the while loop efficiently. To do so, the algorithm maintains, for each $p \in X$, a min-heap Heap_p that contains the values of $\operatorname{avg}(p, C')$ for all clusters $C' \neq C(p)$. Then, to check the condition of the while loop, the algorithm uses Heap_p to find $\min_{C' \neq C(p)} \operatorname{avg}(p, C')$ in time $\tilde{O}(1)$ for each point p. Furthermore, $\operatorname{avg}(p, C(p) \setminus \{p\})$ can be easily computed using the maintained value of $\operatorname{avg}(p, C(p))$ and the equality $\mathrm{avg}(p, C(p) \setminus \{p\}) = \frac{|C(p)|}{|C(p) \setminus \{p\}|} \mathrm{avg}(p, C(p))$, given that the algorithm is also maintaining the size of the clusters. Moreover, once all $\mathrm{avg}(p, C)$ values in lines 2-3 are computed, initializing these heaps, $\{\mathrm{Heap}_p \mid p \in P\}$, require only $\tilde{O}(nk)$ time.

For a swap step, line 7 runs simply in $O(1)$. Moreover, each iteration of the for loop in line 8 can be implemented in time $\tilde{O}(1)$, using the equalities $\mathrm{avg}(p', C \setminus \{p\}) = \frac{1}{|C \setminus \{p\}|} (|C| \,\mathrm{avg}(p, C) - d(p, p'))$ and $\mathrm{avg}(p', C' \cup \{p\}) = \frac{1}{|C' \cup \{p\}|} (|C'| \,\mathrm{avg}(p, C') + d(p, p'))$. Note that the we need to update the min-heaps, which can be done in $\tilde{O}(1)$ since only two elements of each heap have changed.

For a merge and split step, Lemma 2 guarantees that line 14 runs in time $\tilde{O}(n^2)$. Furthermore, it is straightforward to verify that the rest of operations can also be implemented in that time. □

This concludes the analysis of Algorithm 2.

4 Conclusion

In our study of the IP-stable clustering problem, we examined the natural local search algorithm. Notably, recent works (Ahmadi et al., 2022; Aamand et al., 2023) have left open the efficacy of existing natural clustering algorithms concerning the IP-stability metric. We established that by employing a carefully selected update rule, the local search terminates, yielding an $O(\log n)$-IP stable clustering. Moreover, with further refinements, we achieved an algorithm runtime of $\tilde{O}(nk)$, surpassing the runtime of the existing $O(1)$-IP stable clustering by Aamand et al. (2023).

Additionally, we studied IP-stable clustering using alternative functions, including max and median, and provided provable guarantees.

Finally, analyzing a global clustering objective for our proposed local search algorithms remains an interesting open question.

References

A. Aamand, J. Chen, A. Liu, S. Silwal, P. Sukprasert, A. Vakilian, and F. Zhang. Constant approximation for individual preference stable clustering. *Advances in Neural Information Processing Systems*, 36, 2023.

M. Abbasi, A. Bhaskara, and S. Venkatasubramanian. Fair clustering via equitable group representations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*, page 504–514, 2021.

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

S. Ahmadi, P. Awasthi, S. Khuller, M. Kleindessner, J. Morgenstern, P. Sukprasert, and A. Vakilian. Individual preference stability for clustering. In *International Conference on Machine Learning*, pages 197–246, 2022.

S. Ahmadian, A. Epasto, R. Kumar, and M. Mahdian. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 267–275, 2019.

N. Anderson, S. K. Bera, S. Das, and Y. Liu. Distributional individual fairness in clustering. *arXiv preprint arXiv:2006.12589*, 2020.

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*, pages 405–413. PMLR, 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 *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. F. Balcan, T. Dick, R. Noothigattu, and A. D. Procaccia. Envy-free classification. *Advances in Neural Information Processing Systems*, 32, 2019.

S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani. Fair algorithms for clustering. *Advances in Neural Information Processing Systems*, 32, 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)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 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. P. Dickerson, A. Srinivasan, and L. Tsepenekas. Fairness, semi-supervised learning, and more: A general framework for clustering with stochastic pairwise constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

D. Chakrabarti, J. P. Dickerson, S. A. Esmaeili, A. Srinivasan, and L. Tsepenekas. A new notion of individually fair clustering: α-equitable k-center. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

D. Z. Chen, J. Li, H. Liang, and H. Wang. Matroid and knapsack center problems. *Algorithmica*, 75(1): 27–52, 2016.

X. Chen, B. Fain, L. Lyu, and K. Munagala. Proportionally fair clustering. In *International Conference on Machine Learning*, pages 1032–1041, 2019.

F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii. Fair clustering through fairlets. *Advances in Neural Information Processing Systems*, 30, 2017.

A. Chiplunkar, S. Kale, and S. N. Ramamoorthy. How to solve fair k-center in massive data models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1877–1886, 2020.

E. Chlamtáč, Y. Makarychev, and A. Vakilian. Approximating fair clustering with cascaded norm objectives. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2664–2683, 2022.

Z. Dai, Y. Makarychev, and A. Vakilian. Fair representation clustering with several protected classes. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 814–823, 2022.

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

J. H. Dreze and J. Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica*, 48(4):987–1003, 1980.

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

F. Edward Su. Rental harmony: Sperner's lemma in fair division. *The American mathematical monthly*, 106(10):930–942, 1999.

E. Elkind, A. Fanelli, and M. Flammini. Price of pareto optimality in hedonic games. *Artificial Intelligence*, 288:103357, 2020.

D. K. Foley. *Resource allocation and the public sector.* Yale University, 1966.

Y. Gal, M. Mash, A. D. Procaccia, and Y. Zick. Which is the fairest (rent division) of them all? In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 67–84, 2016.

M. Ghadiri, S. Samadi, and S. Vempala. Socially fair k-means clustering. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 438–448, 2021.

M. Ghadiri, M. Singh, and S. S. Vempala. Constant-factor approximation algorithms for socially fair k-clustering. *arXiv preprint arXiv:2206.11210*, 2022.

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

S. S. Hotegni, S. Mahabadi, and A. Vakilian. Approximation algorithms for fair range clustering. In *International Conference on Machine Learning*, pages 13270–13284. PMLR, 2023.

M. Jones, H. Nguyen, and T. Nguyen. Fair k-centers via maximum matching. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 4940–4949, 2020.

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.

D. Kar, M. Kosan, D. Mandal, S. Medya, A. Silva, P. Dey, and S. Sanyal. Feature-based individual fairness in k-clustering. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

R. M. Karp, M. Luby, and N. Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of algorithms*, 10(3):429–448, 1989.

M. Kleindessner, P. Awasthi, and J. Morgenstern. Fair k-center clustering for data summarization. In *36th International Conference on Machine Learning, ICML 2019*, pages 5984–6003. International Machine Learning Society (IMLS), 2019.

R. Krishnaswamy, A. Kumar, V. Nagarajan, Y. Sabharwal, and B. Saha. The matroid median problem. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 1117–1130, 2011.

R. Krishnaswamy, S. Li, and S. Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 646–659, 2018.

E. Laber and L. Murtinho. Optimization of inter-group criteria for clustering with minimum size constraints. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

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)*, pages 3246–3264. PMLR, 2021.

J. Matoušek. Bi-lipschitz embeddings into low-dimensional euclidean spaces. *Commentationes Mathematicae Universitatis Carolinae*, 31(3):589–600, 1990.

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

M. Negahbani and D. Chakrabarty. Better algorithms for individually fair k-clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

A. D. Procaccia. Cake cutting: Not just child's play. *Communications of the ACM*, 56(7):78–87, 2013.

J. Robertson and W. Webb. *Cake-cutting algorithms: Be fair if you can.* CRC Press, 1998.

M. Schmidt, C. Schwiegelshohn, and C. Sohler. Fair coresets and streaming algorithms for fair k-means. In *International Workshop on Approximation and Online Algorithms*, pages 232–251. Springer, 2019.

A.-A. Stoica and C. Papadimitriou. Strategic clustering. 2018.

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

H. R. Varian. Equity, envy, and efficiency. 1973.

M. B. Zafar, I. Valera, M. Rodriguez, K. Gummadi, and A. Weller. From parity to preference-based notions of fairness in classification. *Advances in neural information processing systems*, 30, 2017.

A Proof of Lemma 1

In this section, we prove Lemma 1. Specifically, we prove the following observation, and show that it implies the lemma.

Observation 2. *For every metric space (X, d), every point $p \in X$, and every two non-empty subsets $S_1, S_2 \subseteq X$,*

$$\frac{1}{|S_1| \cdot |S_2|} \sum_{p_1 \in S_1} \sum_{p_2 \in S_2} d(p_1, p_2) \leq \operatorname{avg}(p, S_1) + \operatorname{avg}(p, S_2).$$

Proof. For every metric space (X, d), point $p \in X$, and non-empty sets $S_1, S_2 \subseteq X$, by the triangle inequality,

$$\begin{aligned}\frac{1}{|S_1| \cdot |S_2|} \sum_{p_1 \in S_1} \sum_{p_2 \in S_2} d(p_1, p_2) &\leq \frac{1}{|S_1| \cdot |S_2|} \sum_{p_1 \in S_1} \sum_{p_2 \in S_2} (d(p_1, p) + d(p, p_2)) \\ &= \frac{1}{|S_1|} \sum_{p_1 \in S_1} d(p_1, p) + \frac{1}{|S_2|} \sum_{p_2 \in S_2} d(p, p_2) \\ &= \operatorname{avg}(p, S_1) + \operatorname{avg}(p, S_2).\end{aligned}$$

□

Proof of Lemma 1. By Observation 2,

$$\frac{1}{|S|^2} \sum_{p_1 \in S} \sum_{p_2 \in S} d(p_1, p_2) \leq \operatorname{avg}(p, S) + \operatorname{avg}(p, S).$$

By the definition of $\operatorname{avg}(p_1, S)$ and by the above inequality,

$$\frac{1}{|S|} \sum_{p_1 \in S} \operatorname{avg}(p_1, S) = \frac{1}{|S|^2} \sum_{p_1 \in S} \sum_{p_2 \in S} d(p_1, p_2) \leq \operatorname{avg}(p, S) + \operatorname{avg}(p, S).$$

□

B Proof of Theorem 6

Proof of Theorem 6. As described in Section 2.1, we define the potential function $\Phi(S) \overset{\text{def}}{=} \log |S| \cdot \sum_{p \in S} \operatorname{avg}(p, S)$. Then, our goal is to show that, for every non-empty set $S \subset X$ and every point $p \in X \setminus S$,

$$\operatorname{avg}(p, S) \leq \Phi(S \cup \{p\}) - \Phi(S) \leq 2 \log_2(n) \cdot \operatorname{avg}(p, S). \tag{3}$$

We will take care of the special cases $|S| = 1$ and $|S| = 2$ separately, and then do the more general case $|S| \geq 3$. We note that the constant factor in the value $2 \log n$ could be slightly improved by changing the potential function so that all sets of size at least 3 have a slightly smaller potential, but we choose not to do this for the sake of simplicity.

- **Case $|S| = 1$.** In this case, $S = \{p'\}$ for some point $p' \neq p$. So, we have that $\operatorname{avg}(p, S) = d(p, p')$, that $\Phi(S) = 0$, and that $\Phi(S \cup \{p\}) = \log 2 \cdot (\operatorname{avg}(p', S \cup \{p\}) + \operatorname{avg}(p, S \cup \{p\})) = (\frac{d(p,p')}{2} + \frac{d(p,p')}{2}) = d(p, p')$. Thus $\Phi(S \cup \{p\}) - \Phi(S) = d(p, p') = \operatorname{avg}(p, S)$.

- **Case $|S| = 2$.** In this case, $S = \{p_1, p_2\}$ for some $p_1, p_2 \in X \setminus \{p\}$. So, $\operatorname{avg}(p, S) = \frac{d(p,p_1)+d(p,p_2)}{2}$, and $\Phi(S) = \operatorname{avg}(p_1, S) + \operatorname{avg}(p_2, S) = \frac{d(p_1,p_2)}{2} + \frac{d(p_1,p_2)}{2} = d(p_1, p_2)$. Furthermore,

$$\Phi(S \cup \{p\}) = \log 3 \cdot \sum_{p' \in \{p, p_1, p_2\}} \operatorname{avg}(p', \{p, p_1, p_2\}) = \log 3 \cdot \frac{2}{3} \cdot (d(p, p_1) + d(p, p_2) + d(p_1, p_2)).$$

  Thus, since $\frac{2}{3} \log 3 \geq 1$, we have

$$\Phi(S \cup \{p\}) - \Phi(S) \geq d(p, p_1) + d(p, p_2) = 2 \operatorname{avg}(p, S) \geq \operatorname{avg}(p, S)$$

and since $\frac{2}{3}\log 3 \le \frac{11}{10}$, we have $\Phi(S\cup\{p\}) - \Phi(S) \le \frac{11}{10}(d(p,p_1)+d(p,p_2)) + \frac{1}{10}d(p_1,p_2)$, which, by the triangle inequality means that

$$\Phi(S\cup\{p\}) - \Phi(S) \le \frac{6}{5}(d(p,p_1)+d(p,p_2)) < \log n \cdot (d(p,p_1)+d(p,p_2)) = 2\log n \cdot \mathrm{avg}(p,S)$$

- **Case $|S| \ge 3$.** Let

$$y_S = |S| \quad \text{and} \quad x_S = \sum_{p_1,p_2\in S} d(p_1,p_2) \quad \text{and} \quad x_{S\cup\{p\}} = \sum_{p_1,p_2\in S\cup\{p\}} d(p_1,p_2).$$

Then

$$\Phi(S) = \frac{\log y_S}{y_S} x_S \quad \text{and} \quad \Phi(S\cup\{p\}) = \frac{\log(y_S+1)}{y_S+1} x_{S\cup\{p\}},$$

and

$$x_{S\cup\{p\}} - x_S = \sum_{p_1,p_2\in S\cup\{p\}} d(p_1,p_2) - \sum_{p_1,p_2\in S} d(p_1,p_2) = d(p,p) + 2\sum_{p'\in S} d(p,p') = 2y_S \cdot \mathrm{avg}(p,S).$$

Therefore,

$$\begin{aligned}\Phi(S\cup\{p\}) - \Phi(S) &= \frac{\log(y_S+1)}{y_S+1}\cdot x_{S\cup\{p\}} - \frac{\log y_S}{y_S}\cdot x_S \\ &= \left(\frac{\log(y_S+1)}{y_S+1} - \frac{\log y_S}{y_S}\right) x_S + \frac{\log(y_S+1)}{y_S+1}\cdot 2y_S \cdot \mathrm{avg}(p,S).\end{aligned} \tag{4}$$

Furthermore, since $y_S = |S| \ge 3$ and the function $\frac{\log y}{y}$ is decreasing for $y \ge 3$,

$$\left(\frac{\log(y_S+1)}{y_S+1} - \frac{\log y_S}{y_S}\right) < 0. \tag{5}$$

Lemma 1 exactly says that $x_S \le 2y_S^2 \,\mathrm{avg}(p,S)$. Equation (5) lets us plug $x_S \ge 0$ and $x_S \le 2y_S^2\,\mathrm{avg}(p,S)$ into Equation (4), respectively giving

$$\Phi(S\cup\{p\}) - \Phi(S) \le \frac{\log(y_S+1)}{y_S+1}\cdot 2y_S\cdot \mathrm{avg}(p,S) \le \frac{\log n}{y_S+1}\cdot 2y_S \cdot \mathrm{avg}(p,S) \le 2\log n \cdot \mathrm{avg}(p,S) \tag{6}$$

and

$$\begin{aligned}\Phi(S\cup\{p\}) - \Phi(S) &\ge \left(\frac{\log(y_S+1)}{y_S+1} - \frac{\log y_S}{y_S}\right)\cdot 2y_S^2\,\mathrm{avg}(p,S) + \frac{\log(y_S+1)}{y_S+1}\cdot 2y_S\cdot\mathrm{avg}(p,S) \\ &= \left(\frac{y_S\log(y_S+1)}{y_S+1} - \log y_S\right)\cdot 2y_S\,\mathrm{avg}(p,S) + \frac{\log(y_S+1)}{y_S+1}\cdot 2y_S\cdot\mathrm{avg}(p,S) \\ &= (\log(y_S+1) - \log y_S)\cdot 2y_S\,\mathrm{avg}(p,S) \\ &\ge \frac{\log e}{y_S+1}\cdot 2y_S\,\mathrm{avg}(p,S) \\ &\ge 2\,\mathrm{avg}(p,S),\end{aligned} \tag{7}$$

where the second to last inequality follows from the fact that the derivative of the function $\log x$ in greater than $\frac{\log e}{y_S+1}$ in the whole interval (y_S, y_S+1), and the last inequality follows from the fact that $\frac{y_S}{y_S+1}\log e \ge \frac{3}{4}\log e \ge 1$. Equation (7) and Equation (6) prove the left-hand and right-hand inequalities in Equation (3), as we needed.

□

C Missing Proofs of Section 3

In this section, we prove Lemma 2, Lemma 3, and Lemma 4 from Section 3.

Algorithm 3: Split is the partitioning procedure from Lemma 2.

```
  Data: $(X, d)$ where $|X| = n$, and a $(k-1)$-clustering $\mathcal{C}$.
  Result: a cluster $C^* \in \mathcal{C}$ and a partition of $C^*$ satisfying the condition described in Lemma 2.
1 compute the potential of the clustering $\mathcal{C}$; $\Phi(\mathcal{C}) = \sum_{C \in \mathcal{C}} \Phi(C)$
2 $C^* \leftarrow \arg\max_{\{C | C \in \mathcal{C}, |C|>1\}} \Phi(C)$
3 repeat
4     sample $C_1^* \subset C^*$ of size $\lceil |C^*|/2 \rceil$ chosen uniformly at random
5     $C_2^* \leftarrow C^* \setminus C_1^*$
6 until $\Phi(C_1^*) + \Phi(C_2^*) \leq \Phi(C^*) - \Omega(\frac{\Phi(C^*)}{\log n})$;
7 return $C^*$ and $(C_1^*, C_2^*)$
```

C.1 Proof of Lemma 2

Proof of Lemma 2. The Split algorithm is described in Algorithm 3. The presented algorithm is a Las-Vegas randomized algorithm, but it can be turned into a Monte-Carlo algorithm via standard reductions. We will now analyze this algorithm.

Correctness Analysis. Since the only clusters with non-zero potential have size at least 2, and there are at most k of these, we get that $\Phi(C^*) \geq \frac{1}{k} \sum_{C \in \mathcal{C} \text{ s.t. } |C| \geq 2} \Phi(C) = \frac{1}{k} \Phi(\mathcal{C})$. Thus, since the returned partition must satisfy the inequality from line 5 of the algorithm, it also satisfies $\Phi(C_1^*) + \Phi(C_2^*) \leq \Phi(C^*) - \Omega(\frac{\Phi(\mathcal{C})}{k \log n})$, which proves the correctness of the algorithm.

Time Analysis. Firstly, the potential $\Phi(C)$ of each cluster C can be calculated in time $O(|C|^2) \leq O(n|C|)$, so lines 1 and 2 of the algorithm can be easily implemented in time $O(n^2)$. Furthermore, each iteration of the loop can be performed in time $O(n^2)$, so we just need to show that at most $\tilde{O}(1)$ iterations occur in expectation. This is shown by the following Observation 3. ☐

Observation 3. *In each execution of line 4 of Algorithm 3, the generated partition has a probability of $\tilde{\Omega}(1)$ to satisfy the inequality $\Phi(C_1^*) + \Phi(C_2^*) \leq \Phi(C^*) - \frac{\Phi(C^*)}{4 \log n}$.*

Proof of Observation 3. To prove the observation, it is enough to show that, for a given execution of line 4 of the algorithm,

$$\mathbb{E}\left[\Phi(C_1^*) + \Phi(C_2^*)\right] \leq (1 - \frac{1}{2 \log n}) \cdot \Phi(C^*). \tag{8}$$

To see why this is enough, notice that, by Markov's inequality, Equation (8) implies that

$$\Pr\left[\Phi(C_1^*) + \Phi(C_2^*) \geq (1 - \frac{1}{4 \log n})\Phi(C^*)\right] \leq \frac{\left(1 - \frac{1}{2 \log n}\right)}{\left(1 - \frac{1}{4 \log n}\right)} = \frac{4 \log n - 2}{4 \log n - 1}.$$

Hence,

$$\Pr\left[\Phi(C_1^*) + \Phi(C_2^*) \leq (1 - \frac{1}{4 \log n})\Phi(C^*)\right] \geq 1 - \frac{4 \log n - 2}{4 \log n - 1} = \frac{1}{4 \log n - 1} = \tilde{\Omega}(1).$$

So, our goal for the rest of the proof is to show that, for a given execution of line 4 of the algorithm, Equation (8) holds: For each point $p \in C^*$, let $C_{i(p)}^*$ denote the side of the partition to which p belongs. Now, since $\Phi(C_1^*) = \log |C_1^*| \sum_{p \in C_1^*} \operatorname{avg}(p, C_1^*) = \log(\lceil |C^*|/2 \rceil) \sum_{p \in C_1^*} \operatorname{avg}(p, C_1^*)$ and similarly $\Phi(C_2^*) \leq \log(\lceil |C^*|/2 \rceil) \sum_{p \in C_2^*} \operatorname{avg}(p, C_2^*)$, we get by linearity of expectation that

$$\mathbb{E}\left[\Phi(C_1^*) + \Phi(C_2^*)\right] \leq \log(\lceil |C^*|/2 \rceil) \sum_{p \in C^*} \mathbb{E}\left[\operatorname{avg}(p, C_{i(p)}^*)\right]. \tag{9}$$

Now, for a given point p, consider the process that samples a partition (C_1^*, C_2^*) as in line 4 of the algorithm, and then samples a point p_{sample} from $C_{i(p)}^*$ uniformly at random. Notice that $d(p, p_{\text{sample}})$ can be non-zero only when p_{sample} is a point from $C^* \setminus \{p\}$. Furthermore, by a symmetry argument, each point in $C^* \setminus \{p\}$ has the same probability of being chosen as p_{sample} by the aforementioned process. Thus,

$$\mathbb{E}_{(C_1^*,C_2^*)}\left[\mathbb{E}_{p_{\text{sample}}\sim C_{i(p)}^*}\left[p_{\text{sample}}\right]\right] = \frac{\Pr[p_{\text{sample}} \in C^* \setminus \{p\}]}{|C^* \setminus \{p\}|} \sum_{p' \in C^* \setminus \{p\}} d(p, p').$$

So, since

$$\Pr[p_{\text{sample}} \in C^* \setminus \{p\}] = \mathbb{E}_{(C_1^*,C_2^*)}\left[(|C_{i(p)}^*| - 1)/|C_{i(p)}^*|\right] \le (|C^*| - 1)/|C^*|,$$

we get that

$$\mathbb{E}_{(C_1^*,C_2^*)}\left[\mathbb{E}_{p_{\text{sample}}\sim C_{i(p)}^*}\left[p_{\text{sample}}\right]\right] \le \frac{1}{|C^*|} \sum_{p' \in C^* \setminus \{p\}} d(p, p') = \operatorname{avg}(p, C^*).$$

However, for each such point p and each possible partition (C_1^*, C_2^*), the value of $\mathbb{E}_{p_{\text{sample}}\sim C_{i(p)}^*}[d(p, p_{\text{sample}})]$ is exactly $\operatorname{avg}(p, C_{i(p)}^*)$, so plugging this into the previous inequality gives

$$\forall p \in C^*, \quad \mathbb{E}_{(C_1^*,C_2^*)}\left[\operatorname{avg}(p, C_{i(p)}^*)\right] \le \operatorname{avg}(p, C^*).$$

Plugging this inequality into Equation (9), we get that

$$\mathbb{E}\left[\Phi(C_1^*) + \Phi(C_2^*)\right] \le \log(\lceil |C^*|/2 \rceil) \sum_{p \in C^*} \operatorname{avg}(p, C^*)$$

Furthermore, since $|C^*| \ge 2$, we have that $\lceil |C^*|/2 \rceil \le 2|C^*|/3$ and thus $\log(\lceil |C^*|/2 \rceil) \le \log |C^*| - 1/2$, so we get that

$$\mathbb{E}\left[\Phi(C_1^*) + \Phi(C_2^*)\right] \le (\log |C^*| - 1/2) \cdot \sum_{p \in C^*} \operatorname{avg}(p, C^*) = \frac{\log |C^*| - 1/2}{\log |C^*|} \cdot \Phi(C^*)$$

which proves Equation (8), as we needed. □

C.2 Proof of Lemma 3

Proof of Lemma 3. Firstly, for every point $p \in X$, the inequality

$$\frac{2\log n}{\max\{|C|, |C'|\}} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2) \le 2\log n \cdot \min\{|C|, |C'|\} \cdot (\operatorname{avg}(p, C) + \operatorname{avg}(p, C'))$$

follows easily from Observation 2, so all we need to prove is that the increase in potential due to merging two clusters, C and C', is bounded by

$$\Phi(C \cup C') - \Phi(C) - \Phi(C') \le \frac{2\log n}{\max\{|C|, |C'|\}} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2).$$

We will divide this proof to two cases, based on whether one of the clusters is of size 1.

Case $\min\{|C|, |C'|\} = 1$. For this case, we assume without loss of generality that C' is the cluster of size 1, and let p denote the single point in C'. Then, $\Phi(C') = 0$, and increase in potential can be rewritten as

$$\Phi(C \cup C') - \Phi(C) - \Phi(C') = \Phi(C \cup \{p\}) - \Phi(C)$$

Furthermore, since Φ is the potential function form Theorem 6, that theorem promises that $\Phi(C \cup \{p\}) - \Phi(C) \le 2\log_2(n) \cdot \operatorname{avg}(p, C)$. So,

$$\begin{aligned}\Phi(C \cup C') - \Phi(C) - \Phi(C') \le 2\log n \cdot \operatorname{avg}(p, C) &= \frac{2\log n}{|C|} \sum_{p_1 \in C} d(p_1, p) \\ &= \frac{2\log n}{\max\{|C|, |C'|\}} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2),\end{aligned}$$

as we needed.

Case $\min\{|C|, |C'|\} \geq 2$. For this case, we let $f : \mathbb{N}_{\geq 2} \to \mathbb{R}_{>0}$ denote the function $f(\ell) = \frac{\log \ell}{\ell}$. Then, since $|C \cup C'| \geq 4$, since f is decreasing in the interval $[4, \infty)$, and since $f(2), f(3) \geq f(4)$, it must be that $f(|C|), f(|C'|) \geq f(|C \cup C'|)$. Furthermore, for this function f, the potential of any set S with $|S| \geq 2$ is $\Phi(S) = f(|S|) \sum_{p_1 \in S} \sum_{p_2 \in S} d(p_1, p_2)$. Together, these facts imply that

$$\Phi(C \cup C') = f(|C \cup C'|) \sum_{p_1 \in C \cup C'} \sum_{p_2 \in C \cup C'} d(p_1, p_2) \leq \Phi(C) + \Phi(C') + 2f(|C \cup C'|) \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2),$$

so

$$\Phi(C \cup C') - \Phi(C) - \Phi(C') \leq 2 \frac{\log |C \cup C'|}{|C \cup C'|} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2) \leq \frac{2 \log n}{\max\{|C|, |C'|\}} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2).$$

This concludes the proof of the lemma. □

C.3 Proof of Lemma 4

Proof of Lemma 4. To prove this lemma, we just need to show that, for every k-clustering $\mathcal{C}$ of a metric space (X, d), the k-centers value of $\mathcal{C}$ is a poly(n) approximation for the potential of $\mathcal{C}$. Let $\text{diameter}(\mathcal{C}) \stackrel{\text{def}}{=} \max_{C \in \mathcal{C}} \text{diameter}(C) = \max_{C \in \mathcal{C}} \max_{p, p' \in C} d(p, p')$. It is a known fact that $Diam(\mathcal{C})$ is a 2-approximation for the k-centers value of $\mathcal{C}$, so it is enough if we prove that diameter$(\mathcal{C})$ is a poly(n) approximation for the potential of $\mathcal{C}$. Indeed, since $\Phi(\mathcal{C}) = \sum_{C \in \mathcal{C}} \frac{\log |C|}{|C|} \sum_{p, p' \in C} d(p, p')$, it holds that $\Phi(\mathcal{C}) \geq \frac{1}{n} \text{diameter}(\mathcal{C})$ and that $\Phi(\mathcal{C}) \leq kn^2 \log n \cdot \text{diameter}(\mathcal{C})$.[1] □

D Faster Implementation of Local Search

The goal of this section is to prove Theorem 2, by designing an adaptation of local search that runs in time $\tilde{O}(nk)$. For the sake of simplicity, we will separate the steps of the local search into epochs, where each epoch reduces the potential of the maintained clustering by a constant factor. In Appendix D.2, we formally prove that if such an epoch can be implemented fast, then the local search can be implemented fast. Thus, our main focus is to prove that an epoch can be implement fast, as stated in the following theorem. We prove this theorem in Appendix D.3

Theorem 7. *There exists a Monte-Carlo randomized algorithm* Epoch *(Algorithm 5) that runs in time $\tilde{O}(nk)$, and given a metric space (X, d) with $|X| = n$ and a k-clustering $\mathcal{C}$ of (X, d), computes a k-clustering $\mathcal{C}'$ of (X, d) that is either $O(\log n)$-IP stable or has $\sum_{C \in \mathcal{C}'} \Phi(C) < \frac{3}{4} \cdot \sum_{C \in \mathcal{C}} \Phi(C)$. The error probability can be made as low as $O(\frac{1}{n^c})$, for any predefined constant $c \geq 1$.*

We will now present a sketch of what additional modifications need to be made to the local search strategy from Algorithm 2 in order to make it run in time $\tilde{O}(nk)$. To do this, we focus on a single epoch of the local search, and let Φ_0 be the potential of the clustering at the beginning of the epoch.

As long as $\Phi(\mathcal{C}) \geq \Omega(\Phi_0)$, a cluster split can reduce the potential of the clustering by $\tilde{\Omega}(\Phi_0/k)$, so the strategy from Algorithm 2 either performs a "merge and split" step that reduces $\Phi(\mathcal{C})$ by $\tilde{\Omega}(\Phi_0/k)$ in time $\tilde{O}(n^2)$, or performs a "swap" step that reduces the potential by $\tilde{\Omega}(\Phi_0/(nk))$ in time $\tilde{O}(n)$. In both these operations, the ratio between the time spent and the decrease in potential is $\tilde{O}(n^2k/\Phi_0)$, resulting in the running time of $\tilde{O}(n^2k)$ for Algorithm 2. Furthermore, once $\Phi(\mathcal{C}) < \Omega(\Phi_0)$, we can simply end the current epoch.

In order to reduce the running time by a factor of n, we need to reduce the ratio between the time spent and the decrease in potential to be $\tilde{O}(nk/\Phi_0)$, which means that we need to implement each merge and split step in time $\tilde{O}(n)$, and implement each swap step in time $\tilde{O}(1)$.

Merge and Split Step. In evaluating the runtime of a merge and split step, the primary bottlenecks are the split procedure itself and the time taken to compute the values of avg(p, C) for each new cluster and for every $p \in X$. Moreover, within the split procedure, the bottleneck is calculating the potential of each potential random split to determine if it is a good split. This computation is essentially about finding values of avg(p, C), given

[1] While a tighter bound on $\Phi(\mathcal{C})$ relative to the diameter is attainable, the current bound is sufficient for our purposes.

that $\Phi(\mathcal{C}) = \log(|C|) \cdot \sum_{p \in C} \operatorname{avg}(p, C)$. Thus, to implement a merge and split step quickly, we just need to know how to quickly compute $\operatorname{avg}(p, C)$ for a specified cluster C and all points $p \in C$. It turns out that if we want to compute these values approximately, we can do it via *importance sampling*.

Swap Step. In the swap operation, the bottlenecks are finding the point p that needs to be swapped, and updating $\operatorname{avg}(p', C)$ and $\operatorname{avg}(p', C')$ for all points $p' \in X$. In order to find a candidate point p for the swap operation, the implementation from Section 3.1 was already supporting a min-heap for each point p, which gives the algorithm access to $\min_{C' \in \mathcal{C} \setminus \{C(p)\}} \operatorname{avg}(p, C')$. To speed up this process further and get $\tilde{O}(1)$ runtime, we just need to maintain an additional min-heap that contains the values $\left\{ \frac{\min_{C' \in \mathcal{C} \setminus \{C(p)\}} \operatorname{avg}(p, C')}{\operatorname{avg}(p, C(p) \setminus \{p\})} \right\}_{p \in X}$. The more challenging part is updating the values $\operatorname{avg}(p', C)$ and $\operatorname{avg}(p', C')$ for all points $p' \in X$. Since updating these $\Theta(n)$ different values requires $\Omega(n)$ time regardless of how fast we can compute them, we cannot update them after every swap step. Instead, we show that it is possible to keep track of an upper-bound on the additive difference between the old, outdated, value $\widehat{\operatorname{avg}}(p, C)$ and the true unknown value $\operatorname{avg}(p, C)$, and only recompute these values once $|\widehat{\operatorname{avg}}(p, C) - \operatorname{avg}(p, C)|$ becomes too large.

We further demonstrate that for each $\operatorname{avg}(p', C)$ and $\operatorname{avg}(p', C')$, the increase in this upper-bound due to a swap step is proportionate to the corresponding decrease in $\Phi(\mathcal{C})$. As a result, the cumulative increase in all these upper bounds throughout the entire epoch aligns proportionally with Φ_0.

Finally, we need to ensure that the additive error $|\widehat{\operatorname{avg}}(p, C) - \operatorname{avg}(p, C)|$ remains roughly on par with Φ_0 before recalculating $\widehat{\operatorname{avg}}(p, C)$. To do this, we show that distinguishing between the case p is α-envious of C' and the case p is less than $(\alpha/2)$-envious of C' only requires estimates of $\operatorname{avg}(p, C(p))$ and $\operatorname{avg}(p, C')$ that are correct up to an additive term of $\approx \frac{1}{\alpha}(\operatorname{avg}(p, C(p)) + \operatorname{avg}(p, C'))$. So, by Observation 2, we can tolerate an additive error that is roughly equal to the "distance" $\frac{1}{|C| \cdot |C'|} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2)$. Furthermore, we can ensure that the distance $\frac{1}{|C| \cdot |C'|} \sum_{p_1 \in C} \sum_{p_2 \in C'} d(p_1, p_2)$ is roughly comparable to Φ_0. If not, we can perform a merge and split step that combines the clusters C and C'.

D.1 Preliminaries for the Faster Implementation

Remark Regarding High-Probability Guarantees. In the analysis of all algorithms in Appendix D, we assume that the outputs of the probabilistic subroutines always satisfy the high-probability guarantee. Since the probability of failure in all of these subroutines can be made to be an arbitrarily small polynomial at the cost of only $\operatorname{polylog}(n)$ factors blowup in the running time, and all algorithms in this section run in polynomial time, this assumption is justified. Furthermore, we note that it is possible to deterministically check, in time $O(nk)$, whether the final clustering returned by the algorithm from Theorem 2 satisfies the required IP-stability condition, so the algorithm can be made to be a Las-Vegas algorithm with expected running time $\tilde{O}(nk)$.

Importance Sampling. Importance sampling is a statistical technique used to estimate properties of a particular distribution, while only having samples generated from a different distribution than the distribution of interest. It achieves this by reweighting the samples in such a way that their distribution matches the target distribution, thus enhancing the efficiency of Monte Carlo algorithms, as in (Karp et al., 1989).[2]

Theorem 8 (Importance Sampling (Karp et al., 1989))**.** *Let $x_1, \ldots, x_n \geq 0$ be non-negative values, and let $\hat{x}_1, \ldots, \hat{x}_n \geq 0$ be estimates of $x_1, \ldots, x_n$ such that $x_i \leq \hat{x}_i$ holds for all $i \in [n]$, and $\sum_{i=1}^n \hat{x}_i = O\left(\sum_{i=1}^n x_i\right)$. Furthermore, let $i_1, \ldots, i_t$ be independent samples from the distribution over $\{1, \ldots, n\}$ that gives probability $\frac{\hat{x}_i}{\sum_{i=1}^n \hat{x}_i}$ to each i. Then, $t = O(1/\epsilon^2)$ suffices so that, w.h.p,*

$$(1-\epsilon) \sum_{i=1}^n x_i \leq \frac{\left(\sum_{i=1}^n \hat{x}_i\right)}{t} \sum_{j=1}^t \frac{x_{i_j}}{\hat{x}_{i_j}} \leq (1+\epsilon) \sum_{i=1}^n x_i.$$

By the importance sampling technique, we prove the following lemma. Proof of Lemma 5 is deferred to Appendix D.4

[2]For more details, see this lecture note: https://tinyurl.com/lect-imp-samp

Lemma 5. *Given a metric space (X, d), a non-empty cluster $C \subseteq X$, a set $S \subseteq X$, and $0 < \epsilon \leq 1$,* CalcAverage *(Algorithm 6) runs in time $\tilde{O}(\frac{|C|+|S|}{\epsilon^2})$, and w.h.p., for each $p \in S$, computes a $(1+\epsilon)$-approximation of $\operatorname{avg}(p, C)$.*

More precisely, for each $p \in S$, CalcAverage *returns $\widehat{\operatorname{avg}}(p, C)$ s.t. $\operatorname{avg}(p, C) \leq \widehat{\operatorname{avg}}(p, C) \leq (1+\epsilon)\operatorname{avg}(p, C)$.*

Corollary 1. *There exists an algorithm that runs in time $\tilde{O}(|C|/\epsilon^2)$, and for any non-empty subset $C \in X$, w.h.p. computes a $(1+\epsilon)$-approximation of $\Phi(C)$, where $0 < \epsilon \leq 1$.*

More precisely, for every non-empty $C \subset X$, the algorithm returns $\hat{\Phi}(C)$ s.t. $\Phi(C) \leq \hat{\Phi}(C) \leq (1+\epsilon)\Phi(C)$.

Proof. The algorithm calculates each estimate $\hat{\Phi}(C)$ by calling CalcAverage(C, C, ϵ) and then setting $\hat{\Phi}(C) = \log(|C|) \cdot \sum_{p \in C} \widehat{\operatorname{avg}}(p, C)$, where $\{\widehat{\operatorname{avg}}(p, C)\}_{p \in C}$ are the outputs of CalcAverage. It is straightforward to see that the algorithm runs in time $\tilde{O}(|C|/\epsilon^2)$. Furthermore, since $\Phi(C) = \log(|C|) \cdot \sum_{p \in C} \operatorname{avg}(p, C)$ (see Section 2.1), and the estimates returned by CalcAverage satisfy $\operatorname{avg}(p, C) \leq \widehat{\operatorname{avg}}(p, C) \leq (1+\epsilon)\operatorname{avg}(p, C)$,

$$\Phi(C) \leq \log(|C|) \cdot \sum_{p \in C} \widehat{\operatorname{avg}}(p, C) \leq (1+\epsilon)\Phi(C).$$

Hence, $\Phi(C) \leq \hat{\Phi}(C) \leq (1+\epsilon)\Phi(C)$. □

Corollary 2. *Given a metric space (X, d) with $|X| = n$, a clustering $\mathcal{C}$, and $0 < \epsilon \leq 1$,* CalcPotential *computes a $(1+\epsilon)$-approximation of $\Phi(\mathcal{C}) = \sum_{C \in \mathcal{C}} \Phi(C)$ in time $\tilde{O}(n/\epsilon^2)$ w.h.p.*

More precisely, given a clustering $\mathcal{C}$, CalcPotential *returns an estimate $\hat{\Phi}(\mathcal{C})$ s.t. $\Phi(\mathcal{C}) \leq \hat{\Phi}(\mathcal{C}) \leq (1+\epsilon)\Phi(\mathcal{C})$ holds w.h.p.*

Proof. Simply follows from the guarantee of Corollary 1 for each cluster $C \in \mathcal{C}$. □

Lemma 6. *There exists an algorithm* FastSplit *running in time $\tilde{O}(n)$ that given a metric space (X, d) with n points and a $(k-1)$-clustering $\mathcal{C}$, w.h.p., finds a cluster $C^* \in \mathcal{C}$ and a partition (C_1^*, C_2^*) of C^* such that $\Phi(C_1^*) + \Phi(C_2^*) \leq \Phi(C^*) - \Omega(\frac{\Phi(\mathcal{C})}{k \log n})$.*

Proof. The procedure follows the same meta-algorithm as Split (Algorithm 3), except that now potentials of clusters are computed via the algorithm from Corollary 1 with $\epsilon = \frac{1}{100 \log n}$. Furthermore, the constant in the Ω notation in the condition of the loop is now $\frac{1}{5}$ instead of $\frac{1}{4}$.

Correctness Analysis. When the algorithm returns, the computed estimates must satisfy $\hat{\Phi}(C_1^*) + \hat{\Phi}(C_2^*) \leq \hat{\Phi}(C^*)\left(1 - \frac{1}{5 \log n}\right)$. Therefore, since the estimates are $(1+\epsilon)$-approximations of the true values, with $\epsilon = \frac{1}{100 \log n}$,

$$\Phi(C_1^*) + \Phi(C_2^*) \leq \Phi(C^*) \cdot (1+\epsilon)\left(1 - \frac{1}{5 \log n}\right) \leq \Phi(C^*) \cdot \left(1 - \frac{1}{6 \log n}\right) = \Phi(C^*) - \Omega(\frac{\Phi(C^*)}{\log n})$$

Running Time Analysis. □

Lemma 7. *For every metric space (X, d), every non-empty subset $C \subset X$, every point $p \in X \setminus C$, and every point $p' \in X$,*

$$|\operatorname{avg}(p', C \cup \{p\}) - \operatorname{avg}(p', C)| \leq \operatorname{avg}(p, C)/(|C|+1) = \operatorname{avg}(p, C \cup \{p\})/|C|$$

D.2 Proof of Theorem 2

Now, we show that Theorem 2 can easily be proved using the algorithm Epoch from Theorem 7 as a subroutine.

Proof of Theorem 2. The implementation of Theorem 2 is described in Algorithm 4. Next, we analyze the correctness and running time of the algorithm.

Algorithm 4: Fast-LS runs in time $\tilde{O}(nk)$.

```
  Data: $(X, d)$, $n = |X|$, $k \leq n$, $\alpha = 16 \log n$
  Result: An $\alpha$-IP stable $k$-clustering of $(X, d)$
1 $\mathcal{C}' \leftarrow$ the output clustering by the greedy algorithm of $k$-center (Gonzalez, 1985) on $X$, and $\epsilon \leftarrow 1/10$
2 repeat
3     $\mathcal{C} \leftarrow \mathcal{C}'$, $\mathcal{C}' \leftarrow$Epoch$(\mathcal{C})$
4 until CalcPotential$(\mathcal{C}', \epsilon) \geq \frac{7}{8} \cdot$ CalcPotential$(\mathcal{C}, \epsilon)$;
5 return $\mathcal{C}'$
```

Correctness Analysis. The algorithm only returns a clustering $\mathcal{C}'$ if the condition

$$\text{“}\textsc{CalcPotential}(\mathcal{C}', \epsilon) \geq \frac{7}{8} \cdot \textsc{CalcPotential}(\mathcal{C}, \epsilon)\text{”}$$

is satisfied. Since CalcPotential returns a $(1+\epsilon)$-approximation of the potential of a clustering, with one-sided error, this condition implies that $(1+\epsilon)\Phi(\mathcal{C}') \geq \frac{7}{8} \cdot \Phi(\mathcal{C})$. Since the algorithm always uses $\epsilon = 1/10$, the above condition implies $\Phi(\mathcal{C}') \geq \frac{10}{11} \cdot \frac{7}{8}\Phi(\mathcal{C}) \geq \frac{3}{4} \cdot \Phi(\mathcal{C})$. Therefore, when the above condition is satisfied, because $\mathcal{C}'$ is the clustering returned by a call of Epoch on $\mathcal{C}$, Theorem 7 implies that $\mathcal{C}'$ is an $O(\log n)$-IP stable clustering of (X, d). In summary, a clustering $\mathcal{C}'$ is only returned if it is $O(\log n)$-IP stable.

Time Analysis. Note that the greedy algorithm of k-center algorithm runs in time $O(nk)$. Since, by Theorem 7, Epoch runs in time $\tilde{O}(nk)$, and by Corollary 2, each call to CalcPotential runs in time $\tilde{O}(n)$, we get that each iteration of the loop in Algorithm 4 takes time $\tilde{O}(nk)$. So, it suffices to bound the number of iterations. By Lemma 4, the potential of the initial clustering is at most poly(n) times larger than the minimum possible potential of a k-clustering of (X, d). Furthermore, whenever the condition of the loop is not satisfied, since CalcPotential computes a $(1+\epsilon)$-approximation, we must have $\Phi(\mathcal{C}') < (1+\epsilon) \cdot \frac{7}{8} \cdot \Phi(\mathcal{C}) = \frac{77}{80} \cdot \Phi(\mathcal{C})$. Thus, each iteration except the last one must decrease the potential of the maintained clustering by a constant factor; hence, that there can be at most $O(\log n)$ iterations. So, the total runtime of Algorithm 4 is $\tilde{O}(nk)$. □

D.3 Proof of Theorem 7

In this section, we analyze Algorithm 5, and show that it satisfies the conditions of Theorem 7. Initially, we introduce certain notations to facilitate the algorithm's analysis, which can be found in Definition 4 and Definition 5. Following this, we show that the algorithm is well defined, i.e., it never tries to read uninitialized values (Claim 3). Then, we state some basic invariants that hold during the run of the algorithm (Claim 4), and two additional lemmas that are proven using the invariants (Lemma 8 and Lemma 9). The correctness analysis of the algorithm is in Appendix D.3.5. Prior to analyzing the algorithm's running time, we need to bound the number of various steps the algorithm executes. This is detailed in Appendix D.3.6 and Appendix D.3.7. Then, we do the analysis of the running time in Appendix D.3.8.

Proof of Theorem 7. By Claim 6 and Claim 7, when Algorithm 5 returns a clustering $\mathcal{C}$, it is either α-IP stable, or satisfies $\Phi(\mathcal{C}) < \frac{3}{4} \cdot \Phi(\mathcal{C}_{\text{input}})$. Furthermore, by Corollary 10, the algorithm runs in time $\tilde{O}(nk)$. □

Definition 4 (Step Types)**.** We refer to each iteration of the while loop in line 6 of Algorithm 5 as a "step". We say that an iteration is a swap step if the condition in line 7 was satisfied during that iteration, and otherwise we say that it is a *recompute* step on the cluster C selected in line 20. If the condition in line 28 is satisfied during a recompute step, then we furthermore refer to this step as a merge and split step.

Definition 5. At any point during the execution of Algorithm 5, for every point p and cluster C, we let $\widehat{\text{avg}}(p, C)$ denote the value that was the true value of avg(p, C) during the previous time the algorithm executed line 21 on C. By Lemma 5, $\widetilde{\text{avg}}(p, C) \leq \widehat{\text{avg}}(p, C) \leq (1+\epsilon)\widetilde{\text{avg}}(p, C)$ holds for every $p \in X, C \in \mathcal{C}$ at all times after $\widehat{\text{avg}}(p, C)$ is first initialized.

Claim 3. *At the beginning of every iteration of the loop in line 6 of Algorithm 5, for every cluster $C \in \mathcal{C} \setminus$ ClusterToRecompute, and every point $p \in X$, the values error(C), $\widehat{size}(C)$, numSwaps(C), progress(C), and $\widehat{\text{avg}}(p, C)$ have already been initialized in a previous recompute step on cluster C.*

Algorithm 5: The procedure EPOCH from Theorem 7. It implements a single epoch in the $\tilde{O}(nk)$-time adaptation of the local search. The variables denoted by $progress(C)$ for each $C \in \mathcal{C}$ are introduced solely for the sake of analysis.

```
Data: $(X, d)$, $n = |X|$, $k \leq n$, $\alpha = 16 \log n$, and a $k$-clustering $\mathcal{C}_{input}$ of $X$
Result: A $k$-clustering $\mathcal{C}$ of $(X, d)$ that is either $\alpha$-IP stable or has $\Phi(\mathcal{C}) < (3/4)\Phi(\mathcal{C}_{input})$
1  $\mathcal{C} \leftarrow \mathcal{C}_{input}$
2  $\epsilon \leftarrow 1/10$
3  $\hat{\Phi}(\mathcal{C}) \leftarrow$ CALCPOTENTIAL$(\mathcal{C}, \epsilon)$
4  $t^* \leftarrow \Omega(\frac{\hat{\Phi}(\mathcal{C})}{k \log(n)})/(4 \log n)$
5  $\mathcal{C}_R \leftarrow \mathcal{C}$        ▷ the set of clusters whose average estimates need to be recomputed.
6  while $\mathcal{C}_R \neq \emptyset$ or exists $p \in X$ and $C' \in \mathcal{C} \setminus \{C(p)\}$ s.t. $C(p) \neq \{p\}$ and
     $\frac{|C(p)|}{|C(p) \setminus \{p\}|} \cdot \widehat{\text{avg}}(p, C(p)) > \frac{\alpha}{2} \cdot \widehat{\text{avg}}(p, C')$ do
7      if $\mathcal{C}_R = \emptyset$ then
8          choose $p \in X$ and $C' \in \mathcal{C}$ as in the condition of the while loop
9          move $p$ from $C = C(p)$ to $C'$
10         $progressIncrease \leftarrow (\frac{1}{1+\epsilon}\widehat{\text{avg}}(p, C) - error(C))/2$
11         for $C'' \in \{C, C'\}$ do
12             $error(C'') \leftarrow error(C'') + (\widehat{\text{avg}}(p, C'') + error(C''))/|C''|$
13             $progress(C'') \leftarrow progress(C'') + progressIncrease$
14             $numSwaps(C'') \leftarrow numSwaps(C'') + 1$
15             if $error(C'') > t^*/(100\alpha|C''|)$ or $numSwaps(C'') > \widehat{size}(C'')/2$ then
16                 add $C''$ to $\mathcal{C}_R$
17             end
18         end
19     else
20         $C \leftarrow pop(\mathcal{C}_R)$
21         $\{\widehat{\text{avg}}(p, C)\}_{p \in X} \leftarrow$ CALCAVERAGE$(C, X, \epsilon)$
22         $error(C) \leftarrow 0$
23         $progress(C) \leftarrow 0$
24         $\widehat{size}(C) \leftarrow |C|$, $numSwaps(C) \leftarrow 0$
25         if CALCPOTENTIAL$(\mathcal{C}, \epsilon) < \frac{1+\epsilon}{2} \cdot \hat{\Phi}(\mathcal{C})$ then
26             return $\mathcal{C}$
27         end
28         if exists cluster $C' \in \mathcal{C} \setminus \{C\}$ such that $\frac{\min\{|C|, |C'|\}}{|C'|} \sum_{p \in C'} \widehat{\text{avg}}(p, C) < t^*$ then
29             $C'' \leftarrow C \cup C'$    ▷ merge clusters $C, C'$
30             $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C, C'\}) \cup \{C''\}$
31             $\mathcal{C}_R \leftarrow (\mathcal{C}_R \setminus \{C, C'\}) \cup \{C''\}$
32             FASTSPLIT$(\mathcal{C})$     ▷ a cluster $C^* \in \mathcal{C}$ is split into $C_1^*$ and $C_2^*$ by FASTSPLIT (see Lemma 6)
33             $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C^*\}) \cup \{C_1^*, C_2^*\}$
34             $\mathcal{C}_R \leftarrow (\mathcal{C}_R \setminus \{C^*\}) \cup \{C_1^*, C_2^*\}$
35         end
36     end
37 end
38 return $\mathcal{C}$
```

Claim 4. *At the beginning of every iteration of the loop in line 6 of Algorithm 5, for every cluster $C \in \mathcal{C} \setminus ClustersToRecompute$, the following invariants hold:*

1. *$numSwaps(C) \leq \widehat{size}(C)/2$,*
2. *$error(C) \leq t^*/(100\alpha|C|)$,*
3. *$\left|\widehat{size}(C) - |C|\right| \leq numSwaps(C)$; and,*
4. *for every point $p' \in X$, $|\widetilde{\text{avg}}(p', C) - \text{avg}(p', C)| \leq error(C)$.*

Corollary 3. *At the beginning of every iteration of the loop in line 6 of Algorithm 5, for every cluster $C \in \mathcal{C} \setminus \mathcal{C}_R$, the inequalities $|C|/2 \leq \widehat{size}(C) \leq 2|C|$ hold.*

Proof of Corollary 3. This corollary follows immediately from invariants 1 and 3 of Claim 4 □

Lemma 8. *At the beginning of every iteration of the loop in line 6 of Algorithm 5, for every two different clusters $C, C' \in \mathcal{C} \setminus \mathcal{C}_R$, and every point $p \in X$, the inequalities $\widetilde{\text{avg}}(p, C) + \widetilde{\text{avg}}(p, C') \geq (3/16)\frac{t^*}{\min\{|C|,|C'|\}}$ and $\text{avg}(p, C) + \text{avg}(p, C') \geq (1/8)\frac{t^*}{\min\{|C|,|C'|\}}$ hold.*

Lemma 9. *At every swap step made by Algorithm 5, when the execution is at line 8, the inequality $\text{avg}(p, C(p)) \geq \frac{t^*}{10\min\{|C(p)|,|C'|\}}$ holds.*

D.3.1 Proof of Claim 3

We start by stating and proving the following observation.

Observation 4. *If C belongs to $\mathcal{C} \setminus \mathcal{C}_R$ at the beginning of iteration τ of the loop in line 6 of Algorithm 5, and if this cluster belonged to $\mathcal{C} \cap \mathcal{C}_R$ at the beginning of an earlier iteration $\tau' < \tau$, then there exists some iteration $\tau' \leq \tau'' < \tau$ such that τ'' is a recompute step on cluster C.*

Proof. The only line of the algorithm that might remove a cluster in $\mathcal{C}$ from $\mathcal{C}_R$ is line 20, and clusters that get removed from $\mathcal{C}$ never get inserted into $\mathcal{C}$ again. So, since $C \in \mathcal{C} \setminus \mathcal{C}_R$ holds at the beginning of iteration τ, and $C \in \mathcal{C} \cap \mathcal{C}_R$ held at the beginning of iteration τ', and since clusters that got removed from $\mathcal{C}$ never get reinserted into $\mathcal{C}$, there must be some iteration $\tau' \leq \tau'' < \tau$ such that C was removed from $\mathcal{C}_R$ while being in $\mathcal{C}$ during iteration τ''. So since the only way for a cluster to get removed from $\mathcal{C}_R$ without getting removed from $\mathcal{C}$ is by line 20 of the algorithm, it must be that τ' was a recompute step on cluster C. □

We are now ready to prove Claim 3.

Proof of Claim 3. Let τ be the current iteration, and let τ_C be the first iteration such that C was in $\mathcal{C}$ at the beginning of iteration τ_C. We will begin by proving that C was in the set $\mathcal{C}_R$ at the beginning of iteration τ_C: On the one hand, if τ_C was the first iteration of the algorithm, then C must have been part of the input clustering, which means that C was in the set $\mathcal{C}_R$ at the start of iteration τ_C. On the other hand, if τ_C was not the first iteration of the algorithm, then C must have been added to $\mathcal{C}$ during iteration $\tau_C - 1$. In that case, since the only lines that can add a cluster to $\mathcal{C}$ during an iteration are line 30 and line 33, and since these lines are immediately followed by lines that insert the same cluster into $\mathcal{C}_R$, it must be that C was added to $\mathcal{C}_R$ during iteration $\tau_C - 1$. Furthermore, it is impossible that C was removed from $\mathcal{C}_R$ in iteration $\tau_C - 1$ after being added to it in that same iteration, because the only way that could happen is if it was added by line 30 and then removed by line 33, which would imply that C was removed from $\mathcal{C}$ by line 34 in iteration $\tau_C - 1$, which contradicts the fact that it belongs to $\mathcal{C}$ at the beginning of iteration τ_C. To summarize, we proved that C must have belonged to $\mathcal{C}_R$ at the beginning of iteration τ_C. Therefore, by Observation 4, there must have been an iteration $\tau_C \leq \tau' < \tau$ that was a recompute step on C. □

D.3.2 Proof of Claim 4

Proof of Claim 4. We will prove the claim by induction on the iteration number. The base of the induction holds trivially, since there are no clusters in $\mathcal{C} \setminus \mathcal{C}_R$ at the beginning of the first iteration. So, from now on, we assume the invariants held at the beginning of iteration τ, and our goal is to prove that they hold at the beginning of iteration $\tau + 1$. We divide the proof into three cases based on whether τ is a swap step, a merge and split step, or a recompute step that isn't a merge and split step.

Case: τ is a swap step. In this case, no new clusters were added to the set $\mathcal{C} \setminus \mathcal{C}_R$ during iteration τ, and the only clusters for which any invariant may have become invalidated are the clusters C and C' selected by the swap step. So, we just need to show that the invariants hold for each $C'' \in \{C, C'\}$ such that C'' is still in $\mathcal{C} \setminus \mathcal{C}_R$ at the beginning of iteration $\tau + 1$:

- **invariants 1 and 2.** Since C'' is still in $\mathcal{C} \setminus \mathcal{C}_R$ at the beginning of iteration $\tau + 1$, the condition of the if statement in line 15 of the algorithm must not have been met for this C'', which exactly means that $error(C'') \geq t^*/(100\alpha|C''|)$ and $numSwaps(C'') \leq \widehat{size}(C'')/2$, as stated in the invariants.
- **invariant 3.** Since the invariant $\left|\widehat{size}(C'') - |C''|\right| \leq numSwaps(C'')$ held at the beginning of iteration τ, since $\widehat{size}(C'')$ did not change during iteration τ, since $|C''|$ changed by at most 1 during iteration τ, and since $numSwaps(C'')$ was increased by 1 during iteration τ, the inequality $\left|\widehat{size}(C'') - |C''|\right| \leq numSwaps(C'')$ must still hold at the beginning of iteration $\tau + 1$.
- **invariant 4.** Fix some point $p' \in X$, and we'll show that the invariant holds for this point at the start of iteration $\tau + 1$. Since the invariant held at the beginning of iteration τ, and since $\widetilde{\operatorname{avg}}(p', C'')$ did not change during iteration τ, it is enough if we upperbound the absolute value of the change to $\operatorname{avg}(p', C'')$ during iteration τ by the amount that $error(C'')$ was increased during iteration τ: For each $\tau' \in \{\tau, \tau+1\}$, let $C''_{\tau'}$ and $error(C''_{\tau'})$ denote the state of the cluster C'' and the value of $error(C'')$ at the beginning of iteration τ'. Furthermore, since, for every $p'' \in X$, the values of $\widehat{\operatorname{avg}}(p'', C'')$ and $\widetilde{\operatorname{avg}}(p'', C'')$ did not change during iteration τ, we will write these without a subscript to denote their value during the whole of iteration τ. So, let p be the point selected by line 8 of the algorithm during iteration τ. By Lemma 7, we must have

$$|\operatorname{avg}(p', C'_{\tau+1}) - \operatorname{avg}(p', C'_{\tau})| = |\operatorname{avg}(p', C'_{\tau} \cup \{p\}) - \operatorname{avg}(p', C'_{\tau})| \leq \operatorname{avg}(p, C'_{\tau})/|C'_{\tau+1}|$$

and

$$|\operatorname{avg}(p', C_{\tau}) - \operatorname{avg}(p', C_{\tau+1})| = |\operatorname{avg}(p', C_{\tau+1} \cup \{p\}) - \operatorname{avg}(p', C_{\tau+1})| \leq \frac{\operatorname{avg}(p, C_{\tau+1} \cup \{p\})}{|C_{\tau+1}|} = \frac{\operatorname{avg}(p, C_{\tau})}{|C_{\tau+1}|},$$

which means that $|\operatorname{avg}(p', C''_{\tau}) - \operatorname{avg}(p', C''_{\tau+1})| \leq \operatorname{avg}(p, C''_{\tau})/|C''_{\tau+1}|$ holds regardless of which $C'' \in \{C, C'\}$ we are dealing with. Since the invariant held at the beginning of iteration τ, we know that $\operatorname{avg}(p, C''_{\tau}) \leq \widetilde{\operatorname{avg}}(p, C'') + error(C''_{\tau}) \leq \widehat{\operatorname{avg}}(p, C'') + error(C''_{\tau})$. Together, the last two inequalities imply that $|\operatorname{avg}(p', C''_{\tau}) - \operatorname{avg}(p', C''_{\tau+1})| \leq \left(\widehat{\operatorname{avg}}(p, C'') + error(C''_{\tau})\right)/|C''_{\tau+1}|$. Since $|C''_{\tau+1}|$ is exactly the size of C'' after line 9 is executed in iteration τ, this means that the absolute value of the change to $\operatorname{avg}(p', C'')$ during iteration τ is upperbounded by the amount that $error(C'')$ was increased during iteration τ, as we needed to prove.

Case: τ is a merge and split step. In this case, during step τ, all the clusters that got removed from $\mathcal{C}_R$ also got removed from $\mathcal{C}$, and all the clusters that got added to $\mathcal{C}$ also got added to $\mathcal{C}_R$. There, all clusters that belong to $\mathcal{C} \setminus \mathcal{C}_R$ at the beginning of iteration $\tau + 1$ were also in that set at the beginning of iteration τ. Furthermore, since the cluster C selected at line 20 of the algorithm is one of the clusters that for removed from $\mathcal{C}$ at step τ (line 30), we get that, for every cluster $\overline{C}$ that belongs to $\mathcal{C} \setminus \mathcal{C}_R$ at the beginning of iteration $\tau + 1$, the values of $numSwaps(\overline{C})$, $\hat{size}(\overline{C})$, $error(\overline{C})$, $|\overline{C}|$, $\widetilde{\operatorname{avg}}(p', \overline{C})$, $\operatorname{avg}(p', \overline{C})$ were not changed during step τ, so the invariants all remain.

Case: τ is a recompute step that isn't a merge and split step. Let C be the cluster chosen at line 20 during step τ. Then, for all clusters other than C, it's clear that none of the invariants became invalidated during step τ. So, we just need to prove that the invariants hold for cluster C at the beginning of iteration $\tau + 1$:

- **invariant 1.** Since iteration τ sets $numSwaps(C)$ to 0, this invariant holds.
- **invariant 2.** Since iteration τ sets $error(C)$ to 0, this invariant holds.
- **invariant 3.** Since iteration τ sets $\widehat{size}(C)$ to $|C|$, this invariant holds.
- **invariant 4.** For all $p' \in X$, since the value of $\widetilde{\operatorname{avg}}(p', C)$ at the beginning of iteration $\tau + 1$ is defined as the value of $\operatorname{avg}(p', C)$ during the last recompute step on cluster C that happened before iteration $\tau + 1$ (see Definition 5), and since iteration τ is a recompute step on cluster C, we get $\widetilde{\operatorname{avg}}(p', C)$ at the beginning of iteration $\tau + 1$ is equal to the value of $\operatorname{avg}(p', C)$ at iteration τ. Since the value of $\operatorname{avg}(p', C)$ does not change during iteration τ, this implies that $|\widetilde{\operatorname{avg}}(p', C) - \operatorname{avg}(p', C)| = 0$ holds at the beginning of iteration $\tau + 1$, so the invariant holds at that time.

This concludes the proof of Claim 4. ☐

D.3.3 Proof of Lemma 8

We begin by proving the following claim and corollary.

Claim 5. *At the end of every iteration of the loop in line 6 of Algorithm 5, if this iteration was a recompute step for cluster C, and if at the end of the iteration C is in $\mathcal{C}$, then, for every cluster $C' \in \mathcal{C}$, and every point $p \in X$, the inequality $\operatorname{avg}(p, C) + \operatorname{avg}(p, C') \geq \frac{t^*}{(1+\epsilon)\min\{|C|,|C'|\}}$ holds.*

Proof. At the end of a recompute step on cluster C, if C is still in $\mathcal{C}$, then line 30 of the algorithm must not have been reached in this iteration, which means that the condition of the if statement in line 28 must not have been satisfied. Therefore, for every cluster $C' \in \mathcal{C}$, it must be that the estimated averages $\{\operatorname{avg}(p, C)\}_{p \in X}$ computed in this iteration satisfy $\frac{\min\{|C|,|C'|\}}{|C'|} \sum_{p \in C'} \widehat{\operatorname{avg}}(p, C) \geq t^*$. Since these estimates were computed in this iteration with accuracy $(1 + \epsilon)$, and since true values, as well as the sizes of the clusters, did not change during the iteration, the true values must satisfy $\frac{\min\{|C|,|C'|\}}{|C'|} \sum_{p \in C'} \operatorname{avg}(p, C) \geq t^*/(1 + \epsilon)$ at the end of the iteration. Thus, at the end of the iteration,

$$\frac{t^*}{(1+\epsilon)\min\{|C|,|C'|\}} \leq \frac{1}{|C'|} \sum_{p_1 \in C'} \operatorname{avg}(p, C) = \frac{1}{|C'| \cdot |C|} \sum_{p_1 \in C'} \sum_{p_2 \in C} d(p_1, p_2).$$

So, for every point $p \in X$, by Observation 2,

$$\operatorname{avg}(p, C) + \operatorname{avg}(p, C') \geq \frac{1}{|C'| \cdot |C|} \sum_{p_1 \in C'} \sum_{p_2 \in C} d(p_1, p_2) \geq \frac{t^*}{(1+\epsilon)\min\{|C|,|C'|\}},$$

as we needed to prove. ☐

Corollary 4. *At the beginning of every iteration of the loop in line 6 of Algorithm 5, if the previous iteration was a recompute step for cluster C, and if at the beginning of the current iteration C is in $\mathcal{C} \setminus \mathcal{C}_R$, then, for every cluster $C' \in \mathcal{C} \setminus \mathcal{C}_R$, and every point $p \in X$, the inequality $\widetilde{\operatorname{avg}}(p, C) + \widetilde{\operatorname{avg}}(p, C') \geq (3/8)\frac{t^*}{\min\{\widetilde{size}(C),\widetilde{size}(C')\}}$ holds.*

Proof. Let τ be the current iteration. By Claim 5, $\operatorname{avg}(p, C) + \operatorname{avg}(p, C') \geq \frac{t^*}{(1+\epsilon)\min\{|C|,|C'|\}}$ must hold at the end of the previous iteration, and thus at the start of the current iteration. Thus, by invariants 4 and 2 from Claim 4, at the beginning of the current iteration,

$$\begin{aligned}
\widetilde{\operatorname{avg}}(p, C) + \widetilde{\operatorname{avg}}(p, C') &\geq \frac{t^*}{(1+\epsilon)\min\{|C|,|C'|\}} - error(C) - error(C') \\
&\geq \frac{t^*}{(1+\epsilon)\min\{|C|,|C'|\}} - t^*/(100\alpha|C|) - t^*/(100\alpha|C'|) \\
&\geq \frac{t^*}{(1+\epsilon)\min\{|C|,|C'|\}} - t^*/(50\min\{|C|,|C'|\}) \\
&\geq \frac{3}{4} \cdot \frac{t^*}{\min\{|C|,|C'|\}}.
\end{aligned}$$

So, by Corollary 3, at the beginning of the current iteration, $\widetilde{\text{avg}}(p, C) + \widetilde{\text{avg}}(p, C') \geq (3/8)\frac{t^*}{\min\{\widetilde{size}(C), \widetilde{size}(C')\}}$, as we needed to prove. □

We are now ready to prove the lemma.

Proof of Lemma 8. Let τ be the current iteration, let $p \in X$ be some point, and let C and C' be two different clusters that belong to $\mathcal{C} \setminus \mathcal{C}_R$ at the beginning of iteration τ. We need to prove that, at the beginning of iteration τ, the inequalities $\widetilde{\text{avg}}(p, C) + \widetilde{\text{avg}}(p, C') \geq \frac{t^*}{2\min\{|C|, |C'|\}}$ and $\text{avg}(p, C) + \text{avg}(p, C') \geq \frac{t^*}{4\min\{|C|, |C'|\}}$ hold.

Let $\tau' < \tau$ be the last step before τ that was a recompute step on either of the clusters C and C' (such a step exists by Claim 3). By the definition of $\widetilde{\text{avg}}(p, C)$ and $\widetilde{\text{avg}}(p, C')$ (Definition 5), this means that the values of $\widetilde{\text{avg}}(p, C)$ and $\widetilde{\text{avg}}(p, C')$ did not change between iteration τ' and iteration τ, and by looking at the algorithm, it means that $\widehat{size}(C)$ and $\widehat{size}(C')$ did not change between iteration τ' and iteration τ'. Furthermore, by Corollary 4, at the beginning of iteration $\tau' + 1$, the inequality

$$\widetilde{\text{avg}}(p, C) + \widetilde{\text{avg}}(p, C') \geq (3/8)\frac{t^*}{\min\{\widetilde{size}(C), \widetilde{size}(C')\}}$$

held. So, at the beginning of the current iteration τ, this inequality still holds. Therefore, at the beginning of iteration τ, by Corollary 3,

$$\widetilde{\text{avg}}(p, C) + \widetilde{\text{avg}}(p, C') \geq (3/16)\frac{t^*}{\min\{|C|, |C'|\}},$$

which is the first inequality that we need for the proof of the lemma. Furthermore, by invariants 4 and 2 of Claim 4, the above inequality implies

$$\begin{aligned}\text{avg}(p, C) + \text{avg}(p, C') &\geq \frac{3}{16} \cdot \frac{t^*}{\min\{|C|, |C'|\}} - error(C) - error(C') \\ &\geq \frac{3}{16} \cdot \frac{t^*}{\min\{|C|, |C'|\}} - t^*/(100\alpha|C'|) - t^*/(100\alpha|C|) \\ &\geq \frac{2}{16} \cdot \frac{t^*}{\min\{|C|, |C'|\}},\end{aligned}$$

which is the second inequality that we need for the proof of the lemma. This concludes the proof of Lemma 8. □

D.3.4 Proof of Lemma 9

We begin with the following observation.

Observation 5. *At every swap step made by Algorithm 5, when the execution is at line 8, it must be that* $\widetilde{\text{avg}}(p, C(p)) \geq 2\widetilde{\text{avg}}(p, C')$.

Proof of Observation 5. By the definition of line 8 of the algorithm, the chosen point p and cluster C' satisfy $C(p) \neq \{p\}$ and $(|C(p)|/|C(p) \setminus \{p\}|)\widehat{\text{avg}}(p, C(p)) > (\alpha/2)\widehat{\text{avg}}(p, C')$. Since $C(p) \neq \{p\}$, we have $|C(p) \setminus \{p\}| = |C(p)| - 1 \geq 1$, so $(|C(p)|/|C(p) \setminus \{p\}|) \leq 2$, so the above inequality implies that

$$\frac{\widehat{\text{avg}}(p, C(p))}{\widehat{\text{avg}}(p, C')} \geq \alpha/4.$$

Furthermore, as explained in Definition 5, $\frac{\widehat{\text{avg}}(p, C(p))}{\widehat{\text{avg}}(p, C')} \leq (1+\epsilon)\frac{\widetilde{\text{avg}}(p, C(p))}{\widetilde{\text{avg}}(p, C')}$, so the previous inequality implies that

$$\frac{\widetilde{\text{avg}}(p, C(p))}{\widetilde{\text{avg}}(p, C')} \geq \alpha/(4(1+\epsilon)) \geq 2,$$

which means that $\widetilde{\text{avg}}(p, C(p)) \geq 2\widetilde{\text{avg}}(p, C')$, as we needed to prove. □

We are now ready to prove Lemma 9.

Proof of Lemma 9. Since the step is a swap step, the condition in the if statement at line 7 must have been met, which means that $\mathcal{C}_R$ is empty. So, since the clusters C' and $C = C(p)$ are in $\mathcal{C}$, we get that they are in $\mathcal{C} \setminus \mathit{ClustersToRecompute}$ at the start of this step, so Lemma 8 tells us that $\widetilde{\mathrm{avg}}(p, C) + \widetilde{\mathrm{avg}}(p, C') \geq (3/16)\frac{t^*}{\min\{|C|,|C'|\}}$. Furthermore, by Observation 5, we have $\widetilde{\mathrm{avg}}(p, C) \geq (2/3)(\widetilde{\mathrm{avg}}(p, C) + \widetilde{\mathrm{avg}}(p, C'))$, so the previous inequality gives use that $\widetilde{\mathrm{avg}}(p, C) \geq (2/16)\frac{t^*}{\min\{|C|,|C'|\}} \geq \frac{t^*}{10\min\{|C|,|C'|\}}$, as we needed to prove. $\square$

D.3.5 Correctness Analysis of Algorithm 5

In this section, we prove the correctness of the algorithm, by proving the following Claim 6 and Claim 7

Claim 6. *If the algorithm returns at line 38, then the output is α-IP stable.*

Proof of Claim 6. The proof is by contradiction. So, assume that when the algorithm returned at line 38, there existed some point $p \in X$ and cluster $C' \in \mathcal{C} \setminus \{C(p)\}$ such that $C(p) \neq \{p\}$ and $\mathrm{avg}(p, C(p) \setminus \{p\}) > \alpha \,\mathrm{avg}(p, C')$, and our goal is to show that this leads to a contradiction.

When the algorithm returned, since $C(p) \neq \{p\}$, it must be that $|C(p) \setminus \{p\}| \geq 1$. So, since $\mathrm{avg}(p, C(p) \setminus \{p\}) = \frac{1}{|C(p)\setminus\{p\}|}\sum_{p' \in C(p)} d(p, p') = \frac{|C(p)|}{|C(p)\setminus\{p\}|}\mathrm{avg}(p, C(p))$, the inequality $\mathrm{avg}(p, C(p) \setminus \{p\}) > \alpha \,\mathrm{avg}(p, C')$ gives us

$$\frac{|C(p)|}{|C(p) \setminus \{p\}|}\mathrm{avg}(p, C(p)) > \alpha \,\mathrm{avg}(p, C') \tag{10}$$

and

$$\mathrm{avg}(p, C(p)) > \mathrm{avg}(p, C'). \tag{11}$$

Just before returning in line 38, the algorithm must have exited the while loop in line 6, which means that the condition of that loop was not satisfied. Therefore, $\mathcal{C}_R$ was empty when the algorithm exited the loop, which means that $C(p)$ and C' were both in $\mathcal{C} \setminus \mathcal{C}_R$. Thus, by Lemma 8, $\mathrm{avg}(p, C(p)) + \mathrm{avg}(p, C') \geq (1/8)\frac{t^*}{\min\{|C(p)|,|C'|\}}$ held when the algorithm exited the loop. So, since Equation (11) also held when the algorithm exited the loop, we get that $\mathrm{avg}(p, C(p)) \geq (1/16)\frac{t^*}{\min\{|C(p)|,|C'|\}}$ must have held. Therefore, by invariants 4 and 2 from Claim 4,

$$\begin{aligned}\widetilde{\mathrm{avg}}(p, C(p)) \geq \mathrm{avg}(p, C(p)) - \mathit{error}(C(p)) &\geq \mathrm{avg}(p, C(p)) - t^*/(100\alpha|C(p)|) \\ &\geq \mathrm{avg}(p, C(p)) - \frac{16}{100\alpha}\mathrm{avg}(p, C(p)) \\ &\geq (84/100)\,\mathrm{avg}(p, C(p)),\end{aligned} \tag{12}$$

and

$$\widetilde{\mathrm{avg}}(p, C') \leq \mathrm{avg}(p, C') + \mathit{error}(C') \leq \mathrm{avg}(p, C') + t^*/(100\alpha|C'|) \leq \mathrm{avg}(p, C') + (16/(100\alpha))\,\mathrm{avg}(p, C(p)) \tag{13}$$

held while the algorithm exited the loop. Equation (13) and Equation (10) together give us that

$$\widetilde{\mathrm{avg}}(p, C') < \frac{|C(p)|}{\alpha|C(p) \setminus \{p\}|}\mathrm{avg}(p, C(p)) + (16/(100\alpha))\,\mathrm{avg}(p, C(p)) \leq (116/(100\alpha))\frac{|C(p)|}{|C(p) \setminus \{p\}|}\mathrm{avg}(p, C(p)).$$

So, by Equation (12),

$$\widetilde{\mathrm{avg}}(p, C(p)) \geq \frac{84}{116} \cdot \mathrm{avg}(p, C(p)) > \frac{84}{116} \cdot \alpha \frac{|C(p) \setminus \{p\}|}{|C(p)|} \cdot \widetilde{\mathrm{avg}}(p, C').$$

Therefore, since $\widehat{\mathrm{avg}}(p, C(p))$ and $\widehat{\mathrm{avg}}(p, C')$ are $(1+\epsilon)$-approximations of $\widetilde{\mathrm{avg}}(p, C(p))$ and $\widetilde{\mathrm{avg}}(p, C')$ (see Definition 5), we get that

$$\widehat{\mathrm{avg}}(p, C(p)) > \frac{84}{116} \cdot \alpha \cdot \frac{|C(p) \setminus \{p\}|}{|C(p)|} \cdot \widetilde{\mathrm{avg}}(p, C')$$

held while the algorithm exited the loop, which means that

$$\frac{|C(p)|}{\alpha|C(p) \setminus \{p\}|} \cdot \widehat{\mathrm{avg}}(p, C(p)) > (\alpha/2)\widehat{\mathrm{avg}}(p, C').$$

Since $C(p) \neq \{p\}$ and $C' \in \mathcal{C} \setminus \{C(p)\}$, and the above inequality implies that the condition of the while loop at line 6 of the algorithm was satisfied while the algorithm exited that loop, which is a contradiction.

To summarize, we assumed that the algorithm returned at line 38 while there existed some point $p \in X$ and cluster $C' \in \mathcal{C}$ such that $C(p) \neq \{p\}$ and $\operatorname{avg}(p, C(p) \setminus \{p\}) > \alpha \operatorname{avg}(p, C')$, and got a contradiction. Therefore, if the algorithm returns at line 38, there must not exist such p and C', which exactly means that the returned clustering is α-IP stable. This concludes the proof of Claim 6. $\square$

Claim 7. *If the algorithm returns at line 26, then the returned cluster $\mathcal{C}$ has $\Phi(\mathcal{C}) < \frac{3}{4}\Phi(\mathcal{C}_{\text{input}})$, where $\mathcal{C}_{\text{input}}$ denote the clustering that the algorithm received as input.*

D.3.6 Number of Swap Steps and Merge and Split Steps

In this section, we bound the total number of swap steps, as well as the total number of merge and split step, that may occur during the execution of Algorithm 5. For further details, refer to Corollary 6 and Definition 4.

Claim 8. *At every swap step made by Algorithm 5, when the execution is at line 8, it must be that* $\operatorname{avg}(p, C(p) \setminus \{p\}) > 4\log(n) \cdot \operatorname{avg}(p, C')$.

Proof of Claim 8. By the definition of line 8, the chosen point p and cluster C' satisfy $C(p) \neq \{p\}$ and $(|C(p)|/|C(p) \setminus \{p\}|)\widehat{\operatorname{avg}}(p, C(p)) > (\alpha/2)\widehat{\operatorname{avg}}(p, C')$. Since the values $\widehat{\operatorname{avg}}(p, C(p))$ and $\widehat{\operatorname{avg}}(p, C')$ are $(1+\epsilon)$-approximations of $\widetilde{\operatorname{avg}}(p, C(p))$ and $\widetilde{\operatorname{avg}}(p, C')$ (see Definition 5), the aforementioned inequality implies that

$$(|C(p)|/|C(p) \setminus \{p\}|)\widetilde{\operatorname{avg}}(p, C(p)) > (\alpha/(2+2\epsilon))\widetilde{\operatorname{avg}}(p, C'). \tag{14}$$

Furthermore, when the execution is at line 8, by Lemma 9, $\operatorname{avg}(p, C(p)) \geq \frac{t^*}{10\min\{|C(p)|, |C'|\}}$. So, by invariants 4 and 2 of Claim 4,

$$\begin{aligned}\widetilde{\operatorname{avg}}(p, C(p)) \leq \operatorname{avg}(p, C(p)) + \mathit{error}(C(p)) &\leq \operatorname{avg}(p, C(p)) + t^*/(100\alpha|C(p)|) \\ &\leq \operatorname{avg}(p, C(p)) + (1/(10\alpha)) \operatorname{avg}(p, C(p)) \\ &\leq (11/10) \operatorname{avg}(p, C(p)),\end{aligned} \tag{15}$$

and

$$\begin{aligned}\operatorname{avg}(p, C') \leq \widetilde{\operatorname{avg}}(p, C') + \mathit{error}(C') \leq \widetilde{\operatorname{avg}}(p, C') + t^*/(100\alpha|C'|) &\leq \widetilde{\operatorname{avg}}(p, C') + (1/(10\alpha)) \operatorname{avg}(p, C(p)) \\ &\leq \widetilde{\operatorname{avg}}(p, C') + (1/(10\alpha)) \operatorname{avg}(p, C(p) \setminus \{p\}).\end{aligned} \tag{16}$$

By Equation (14),

$$(|C(p) \setminus \{p\}|/|C(p)|)\frac{\widetilde{\operatorname{avg}}(p, C')}{\widetilde{\operatorname{avg}}(p, C(p))} < \frac{2(1+\epsilon)}{\alpha},$$

so, by Equation (15),

$$(|C(p) \setminus \{p\}|/|C(p)|)\frac{\widetilde{\operatorname{avg}}(p, C')}{\operatorname{avg}(p, C(p))} \leq (|C(p) \setminus \{p\}|/|C(p)|)\frac{\widetilde{\operatorname{avg}}(p, C')}{(10/11)\widetilde{\operatorname{avg}}(p, C(p))} < (11/10)\frac{2(1+\epsilon)}{\alpha},$$

which exactly means that

$$\frac{\widetilde{\operatorname{avg}}(p, C')}{\operatorname{avg}(p, C(p) \setminus \{p\})} < (11/10)\frac{2(1+\epsilon)}{\alpha}.$$

So, by Equation (16),

$$\frac{\operatorname{avg}(p, C')}{\operatorname{avg}(p, C(p) \setminus \{p\})} \leq \frac{\widetilde{\operatorname{avg}}(p, C')}{\operatorname{avg}(p, C(p) \setminus \{p\})} + (1/(10\alpha)) < (11/10)\frac{2(1+\epsilon)}{\alpha} + (1/(10\alpha)) \leq 4/\alpha.$$

Since $\alpha = 16\log n$, the above inequality exactly say that $\operatorname{avg}(p, C(p) \setminus \{p\}) > 4\log(n) \cdot \operatorname{avg}(p, C')$, as we needed. This concludes the proof of Claim 8. $\square$

Corollary 5. *Each swap step reduces the potential $\Phi(\mathcal{C})$ by more than $\tilde{\Omega}(\hat{\Phi}(\mathcal{C})/(nk))$.*

Proof. By Theorem 6, the potential function Φ satisfies that 9 of the algorithm reduces the potential of the cluster C by at least $\mathrm{avg}(p, C \setminus \{p\})$ and increases the potential of the cluster C' by at most $2\log(n) \cdot \mathrm{avg}(p, C')$. By Claim 8, this increase to the potential of C' is less than $\frac{1}{2}\mathrm{avg}(p, C \setminus \{p\})$, so the total change to $\Phi(\mathcal{C})$ is a reduction by more than $\frac{1}{2}\mathrm{avg}(p, C \setminus \{p\})$. Furthermore, by the fact that $\mathrm{avg}(p, C \setminus \{p\}) \geq \mathrm{avg}(p, C)$ and by Lemma 9,

$$\frac{1}{2}\mathrm{avg}(p, C \setminus \{p\}) \geq \frac{1}{2}\mathrm{avg}(p, C) \geq \frac{t^*}{20\min\{|C(p)|, |C|\}} \geq \frac{t^*}{20n}.$$

In summary, each swap step reduce the potential $\Phi(\mathcal{C})$ by at least $\frac{t^*}{20n}$, which is $\tilde{\Omega}(\hat{\Phi}(\mathcal{C})/(nk))$ as we needed. (See the definition of t^* at line 4 of the algorithm.) ☐

Claim 9. *Each merge and split step reduces the potential of the clustering by more than $\tilde{\Omega}(\hat{\Phi}(\mathcal{C})/k)$. I.e., $\Phi(\mathcal{C}_{\text{after}}) < \Phi(\mathcal{C}_{\text{before}}) - \tilde{\Omega}(\hat{\Phi}(\mathcal{C}_{\text{before}})/k)$.*

Proof. Fix a merge and split step during the execution of the algorithm, which is defined as a step where the algorithm enters into the if statement at line 28. (See Definition 4.) Let $\mathcal{C}$ denote the start of the clustering at the beginning of the step, let $\mathcal{C}'$ denote the state of the clustering after the algorithm executed line 30 during this step, and let $\mathcal{C}''$ denote the state of the clustering at the end of the step (i.e. after line 33). Our goal is to show that $\Phi(\mathcal{C}'') < \Phi(\mathcal{C}) - \tilde{\Omega}(\hat{\Phi}(\mathcal{C})/k)$. We begin by separately analyzing the merge and the split, in order to relate $\Phi(\mathcal{C}')$ to $\Phi(\mathcal{C})$, and relate $\Phi(\mathcal{C}'')$ to $\Phi(\mathcal{C}')$,

Separately Analyzing the Merge and the Split. Since this is a merge and split step, the condition of the if statement at line 28 must be satisfied when the algorithm reaches it. So, when the algorithm reached this line, it must be that $\frac{\min\{|C|,|C'|\}}{|C'|}\sum_{p\in C'}\widehat{\mathrm{avg}}(p, C) < t^*$. Furthermore, this happens just after the estimates $\{\widehat{\mathrm{avg}}(p, C)\}_{p\in C}$ are computed in line 21 of the algorithm. So, since the estimates produced by the subroutine CalcAverages can only err from above, the inequality $\frac{\min\{|C|,|C'|\}}{|C'|}\sum_{p\in C'}\mathrm{avg}(p, C) < t^*$ must hold when the algorithm reaches line 28 in this step. So,

$$\begin{aligned} t^* > \frac{\min\{|C|,|C'|\}}{|C'|}\sum_{p\in C'}\mathrm{avg}(p, C) &= \frac{\min\{|C|,|C'|\}}{|C'||C|}\sum_{p_1\in C'}\sum_{p_2\in C}\mathrm{avg}(p_1, p_2) \\ &= \frac{1}{\max\{|C|,|C'|\}}\sum_{p_1\in C'}\sum_{p_2\in C}\mathrm{avg}(p_1, p_2) \end{aligned}$$

must hold at that time. So, by Lemma 3,

$$\Phi(\mathcal{C}') < \Phi(\mathcal{C}) + \log(n)\cdot t^*. \tag{17}$$

Furthermore, by the guarantee of the subroutine FastSplit (see Lemma 6),

$$\Phi(\mathcal{C}'') \leq \Phi(\mathcal{C}') - \Omega(\frac{\Phi(\mathcal{C}')}{k\log(n)}). \tag{18}$$

Putting it Together. Since the algorithm did not return in line 26 during this iteration, it must be that CalcPotential$(\mathcal{C}, \epsilon)$ returned a value which is greater or equal to $((1+\epsilon)/2)\hat{\Phi}(\mathcal{C})$. Therefore, it must be that $(1+\epsilon)\Phi(\mathcal{C}) \geq ((1+\epsilon)/2)\hat{\Phi}(\mathcal{C})$ (see Corollary 2), which means that

$$\Phi(\mathcal{C}) \geq \frac{\hat{\Phi}(\mathcal{C})}{2}. \tag{19}$$

Now, if $\Phi(\mathcal{C}') < \Phi(\mathcal{C}) - \hat{\Phi}(\mathcal{C})/100$, then by Equation (18) we also have $\Phi(\mathcal{C}'') < \Phi(\mathcal{C}) - \hat{\Phi}(\mathcal{C})/100$, in which case we are done. So, for the rest of the analysis, we assume that $\Phi(\mathcal{C}') \geq \Phi(\mathcal{C}) - \hat{\Phi}(\mathcal{C})/100$. By this assumption and by, Equation (19), $\Phi(\mathcal{C}') \geq \Phi(\mathcal{C}) - \Phi(\mathcal{C})/50 = (49/50)\Phi(\mathcal{C})$, which, by Equation (19), means

$$\Phi(\mathcal{C}') \geq \frac{49}{100}\cdot\hat{\Phi}(\mathcal{C}).$$

Therefore, by Equation (18),

$$\Phi(\mathcal{C}'') \leq \Phi(\mathcal{C}') - \frac{49}{100} \cdot \Omega(\frac{\hat{\Phi}(\mathcal{C})}{k\log(n)}),$$

so, by Equation (17),

$$\Phi(\mathcal{C}'') < \Phi(\mathcal{C}) + \log(n) \cdot t^* - \frac{49}{100} \cdot \Omega(\frac{\Phi(\mathcal{C})}{k\log(n)}).$$

Therefore, if we set the constant in the Ω notation of the definition of t^* to the same as the one in the Ω notation of the above inequality (which is the same as the one in the guarantee from Lemma 6), then we get that

$$\Phi(\mathcal{C}'') < \Phi(\mathcal{C}) - \frac{24}{100} \cdot \Omega(\frac{\Phi(\mathcal{C})}{k\log(n)}),$$

which is what we needed to prove. This concludes the proof of Claim 9. □

Corollary 6. *There are at most $\tilde{O}(nk)$ swap steps, and at most $\tilde{O}(k)$ merge-and-split steps.*

Proof of Corollary 6. At the beginning of the algorithm, it set $\hat{\Phi}(\mathcal{C})$ to a value which is at least as large as the initial value of $\Phi(\mathcal{C})$. (See line 3 and Corollary 2.) Furthermore, since the only types of steps which affect the clustering are swap steps and merge and split steps (see Definition 4), Corollary 5 and Claim 9 imply that the potential of the maintained clustering $\mathcal{C}$ never decreases. So, since the potential of clustering can never be negative (see Section 2.1), Corollary 5 and Claim 9 imply that there can be at most $\tilde{O}(nk)$ swap steps, and at most $\tilde{O}(k)$ merge-and-split steps, as we needed to prove. □

D.3.7 Number of Recompute Steps

In this section, we bound the total number of recompute steps that may occur during the execution of Algorithm 5. (See Corollary 9 and Definition 4.)

Claim 10. *whenever lines 12-14 are executed, if x is the amount by which line 13 increases $progress(C)$, then the amount by which line 12 increases $error(C)$ is at most $O(x/\widehat{size}(C''))$, and the amount by which line 14 increases $numSwaps(C'')$ is at most $O(x \cdot \widehat{size}(C'')/t^*)$.*

Proof of Claim 10. Firstly, when the algorithm is at line 8, by Lemma 9, $\text{avg}(p, C(p)) \geq \frac{t^*}{10\min\{|C(p)|,|C'|\}}$. So, by invariants 4 and 2 of Claim 4,

$$\widetilde{\text{avg}}(p, C(p)) \geq \frac{t^*}{10\min\{|C(p)|, |C'|\}} - error(C(p)) \geq \frac{t^*}{10\min\{|C(p)|, |C'|\}} - t^*/(100\alpha|C(p)|)$$
$$\geq \frac{t^*}{11\min\{|C(p)|, |C'|\}}.$$

Therefore, when the algorithm is at line 8, by the fact that $\widehat{\text{avg}}(p, C(p)) \geq \widetilde{\text{avg}}(p, C(p))$ (see Definition 5),

$$\widehat{\text{avg}}(p, C(p)) \geq \frac{t^*}{11\min\{|C(p)|, |C'|\}}$$

which, by invariant 2 of Claim 4, means that

$$\forall C'' \in \{C(p), C'\}, \qquad error(C'') \leq \widehat{\text{avg}}(p, C(p))/(9\alpha) \tag{20}$$

and by Corollary 3, means that

$$\forall C'' \in \{C(p), C'\}, \qquad 1 \leq 11\widehat{\text{avg}}(p, C(p)) \cdot \widehat{size}(C'')/t^* \tag{21}$$

Equation (20) implies that when the algorithm is at line 8, we have

$$\frac{1}{1+\epsilon} \cdot \widehat{\text{avg}}(p, C(p)) - error(C(p)) \geq \frac{1}{2} \cdot \widehat{\text{avg}}(p, C(p))$$

which means that after line 10,

$$progressIncrease \geq \frac{1}{4}\widehat{\text{avg}}(p, C(p)). \tag{22}$$

By Observation 5, when the algorithm is at line 8, $\widetilde{\text{avg}}(p, C(p)) \geq 2\widetilde{\text{avg}}(p, C')$. So, since $\widehat{\text{avg}}(p, C(p))$ and $\widehat{\text{avg}}(p, C')$ are $(1+\epsilon)$-approximations of $\widetilde{\text{avg}}(p, C(p))$ and $\widetilde{\text{avg}}(p, C')$ (see Definition 5), we get that $\widehat{\text{avg}}(p, C(p)) \geq 2\widehat{\text{avg}}(p, C')$. Therefore, at this point, for each $C'' \in \{C(p), C'\}$, by also using invariant 2 of Claim 4,

$$(\widehat{\text{avg}}(p, C'') + error(C''))/\widehat{size}(C'') \leq (\widehat{\text{avg}}(p, C) + error(C''))/\widehat{size}(C'')$$

So, by Equation (20),

$$(\widehat{\text{avg}}(p, C'') + error(C''))/\widehat{size}(C'') \leq (\widehat{\text{avg}}(p, C(p)) + error(C''))/\widehat{size}(C'') \leq \frac{10}{9}\widehat{\text{avg}}(p, C(p))/\widehat{size}(C'')$$

Therefore, by Corollary 3, when the algorithm increases $error(C'')$ at line 12, it increases it by at most $\frac{20}{9}\widehat{\text{avg}}(p, C(p))/\widehat{size}(C'')$. Furthermore, by Equation (22), line 13 increases $error(C'')$ by at least $\frac{1}{4}\widehat{\text{avg}}(p, C(p))$.

In summary, line 12 increases $error(C'')$ by at most $\frac{20}{9}\widehat{\text{avg}}(p, C(p))/\widehat{size}(C'')$, while line 13 increases $progress(C'')$ by at least $\frac{1}{4}\widehat{\text{avg}}(p, C(p))$, and line 14 increases $numSwaps(C'')$ by at most $11\widehat{\text{avg}}(p, C(p)) \cdot \widehat{\text{avg}}(p, C(p))/t^*$ (Equation (21)). This concludes the proof of Claim 10. □

Claim 11. *During line 15, for every cluster $C \in \mathcal{C}$, $progress(C) = \Omega(error(C)\cdot\widehat{size}(C))$ and also $progress(C) = \Omega(numSwaps(C) \cdot t^*/\widehat{size}(C))$.*

Proof of Claim 11. Let τ be the current iteration. Since we reached line 15 in this iteration, it must have been a swap step. By the condition of the if statement in line 7, $\mathcal{C}_R$ must have been empty at the beginning of the iteration, which means that $C \in \mathcal{C} \setminus \mathcal{C}_R$ held. Therefore, by Claim 3, there was some recompute step on the cluster C before step τ. Let τ' be the last such recompute step. At step τ', lines 22 and 23 set $error(C)$ and $progress(C)$ and $numSwaps(C)$ to 0. After that, until the end of iteration τ, the only times were $error(C)$, $progress(C)$, and $numSwaps(C)$, were changed are by lines 12, 13, and 14 during swap steps. So, by Claim 10, we get that $error(C) = O(progress(C)/\widehat{size}(C))$ and $numSwaps(C) = O(progress(C) \cdot \widehat{size}(C)/t^*)$ hold at the beginning of iteration τ, which exactly means that $progress(C) = \Omega(error(C) \cdot \widehat{size}(C))$ and $progress(C) = \Omega(numSwaps(C) \cdot t^*/\widehat{size}(C))$, as we needed. □

Corollary 7. *Whenever a cluster C is inserted into $\mathcal{C}_R$ by line 16, it must have $progress(C) = \tilde{\Omega}(t^*) = \tilde{\Omega}(\hat{\Phi}(\mathcal{C})/k)$.*

Proof of Corollary 7. This follows directly from Claim 11 and from the condition of the if statement in line 15. □

Claim 12. *For every cluster C'', whenever $progress(C'')$ is increased by a swap step, this swap step must have reduced $\Phi(\mathcal{C})$ by a greater amount.*

Proof. Let Δ be the amount by which the current swap step reduces $\Phi(\mathcal{C})$. By Theorem 6, the potential function Φ satisfies that line 9 of the algorithm reduces the potential of the cluster $C(p)$ by at least $\text{avg}(p, C(p) \setminus \{p\})$ and increases the potential of the cluster C' by at most $2\log(n) \cdot \text{avg}(p, C')$. By Claim 8, this increase to the potential of C' is less than $\frac{1}{2}\,\text{avg}(p, C(p) \setminus \{p\})$, so the total change to $\Phi(\mathcal{C})$ is a reduction by more than $\frac{1}{2}\,\text{avg}(p, C(p) \setminus \{p\})$. Therefore, since $\text{avg}(p, C(p) \setminus \{p\}) \geq \text{avg}(p, C(p))$, we get that

$$\Delta > \frac{1}{2}\,\text{avg}(p, C(p)). \tag{23}$$

At the beginning of the current swap step, by the fact that $\widehat{\text{avg}}(p, C(p))$ is a $(1+\epsilon)$ approximation of $\widetilde{\text{avg}}(p, C(p))$ (see Definition 5), together with invariant 4 of Claim 4, and with Equation (23),

$$(\frac{1}{1+\epsilon}\widehat{\text{avg}}(p, C(p)) - error(C(p)))/2 \leq (\widetilde{\text{avg}}(p, C(p)) - error(C(p)))/2 \leq (\text{avg}(p, C(p)))/2 < \Delta.$$

Therefore, line 10 set the variable $ProgressIncrease$ to less than Δ, which means that for each C'', the amount by which the current swap step increases $progress(C'')$ is less than x (i.e. less than the amount by which the current swap step reduces $\Phi(\mathcal{C})$), as we needed to prove. □

Corollary 8. *Throughout the execution of Algorithm 5, line 16 is executed at most $\tilde{O}(k)$ times.*

Proof of Corollary 8. By Corollary 7, whenever line 16 is executed on a cluster C'', this cluster must have $progress(C'') \geq \tilde{\Omega}(\hat{\Phi}(\mathcal{C})/k)$. Furthermore, by Claim 12, this implies that since the last time a recompute step happened on cluster C'', the cluster C'' has been involved in swap steps that reduced $\Phi(\mathcal{C})$ by a total greater than $\tilde{\Omega}(\hat{\Phi}(\mathcal{C})/k)$. Since a recompute step on cluster C'' has to happen between any two consecutive times that line 16 is executed on a cluster C'', this implies that, for any one cluster C, if line 16 has been executed on this cluster C a total of x times throughout the algorithm, then C has been involved in swap steps that reduced $\Phi(\mathcal{C})$ by a total greater than $\tilde{\Omega}(x\hat{\Phi}(\mathcal{C})/k)$. So, since each swap step can involve at most two clusters, we get that the total reduction to $\Phi(\mathcal{C})$ throughout the algorithm is greater than $(\sum_C x_C) \cdot \tilde{\Omega}(\hat{\Phi}(\mathcal{C})/k)/2$, where $(\sum_C x_C)$ denotes the total number of times that line 16 has been executed throughout the algorithm. Therefore, since the value of $\hat{\Phi}(\mathcal{C})$ is at least as large as the initial value of $\Phi(\mathcal{C})$ (see line 3 and Corollary 2), we get that the total number of times that line 16 has been executed throughout the algorithm is at most $\hat{\Phi}(\mathcal{C})/\tilde{\Omega}(\hat{\Phi}(\mathcal{C})/k) = \tilde{O}(k)$. □

Corollary 9. *Throughout the execution of Algorithm 5, there are at most $\tilde{O}(k)$ recompute steps.*

Proof of Corollary 9. Each recompute step removes one cluster from $\mathcal{C}_R$ at line 20. Furthermore, $\mathcal{C}_R$ is created with k clusters, and the only lines that insert clusters into $\mathcal{C}_R$ are line 16, line 31, and line 34, which each insert a constant number of clusters each time they are executed. So, since Corollary 8 says that line 16 is executed at most $\tilde{O}(k)$ times throughout the execution of the algorithm, and Corollary 6 implies that lines 31 and 34 are executed at most $\tilde{O}(k)$ times throughout the execution of the algorithm, we get that the total number of recompute step throughout the execution of the algorithm can be at most $k + O(1) \cdot \tilde{O}(k) = \tilde{O}(k)$. □

D.3.8 Running Time Analysis of Algorithm 5

In this section, we bound the total running time of Algorithm 5. We begin by proving the following two claims, and then complete the analysis in Corollary 10.

Claim 13. *Each swap step can be implemented in time $\tilde{O}(1)$.*

Proof of Claim 13. To quickly implement the checking of the condition of the loop in line 6, and to implement line 8, the algorithm maintains, for each $p \in X$, a min-heap Heap_p that contains the values of $\widehat{avg}(p, C')$ for all clusters $C' \in \mathcal{C} \setminus \{C(p)\}$, where each value is accompanied by a pointer to the respective cluster C'. Furthermore, the algorithm maintains a min-heap MainHeap that contains $\frac{\min_{C' \in \mathcal{C} \setminus \{C(p)\}} \widehat{\text{avg}}(p,C')}{(|C(p)|/|C(p) \setminus \{p\}|)\widehat{\text{avg}}(p,C(p))}$ for each $p \in X$ such that $C(p) \neq \{p\}$ and $\widehat{\text{avg}}(p, C(p)) \neq 0$, where each value is accompanied by a pointer to the respective point p. These min-heaps while only increasing the running time of the algorithm by a multiplicative $\tilde{O}(1)$, where the implementation of the updates to MainHeap uses the other heaps to find the appropriate $\min_{C' \in \mathcal{C} \setminus \{C(p)\}} \widehat{\text{avg}}(p, C')$ for each point. Furthermore, using these heaps, it's clear that checking of the condition of the loop in line 6, and implementing line 8, can both be done in time $\tilde{O}(1)$. Since the rest of the lines executed during a swap step can all be trivially executed in time $\tilde{O}(1)$, we get that a swap step runs in time $\tilde{O}(1)$, as we needed to prove. □

Claim 14. *Each recompute step (including merge and split steps) can be implemented in time $\tilde{O}(n)$*

Proof of Claim 14. It is already explained in the proof of Claim 13 how the condition of the while loop in line 6 can checked in time $\tilde{O}(1)$. Furthermore, by Lemma 5, Corollary 2, and Lemma 6, the procedures CalcAverage, CalcPotential, and FastSplit each run in time $\tilde{O}(n)$. This implies that each recompute step runs in time $\tilde{O}(n)$, as we needed to prove. □

Corollary 10. *The total running time of Algorithm 5 is $\tilde{O}(nk)$.*

Proof of Corollary 10. Claim 13 and Corollary 6 imply that the total time spent on swap steps is $\tilde{O}(nk)$, while Claim 14 and Corollary 9 imply that the total time spent on recompute steps is $\tilde{O}(nk)$. Therefore, the total time that the algorithm spends in the while loop at line 6 is $\tilde{O}(nk)$. Furthermore, since CalcPotential runs in time $\tilde{O}(n)$ (see Corollary 2, the total amount of time that the algorithm spends outside of this loop is $\tilde{O}(n)$. This concludes the proof of Corollary 10. ☐

D.4 Approximating Average Distances Fast

The goal of this section is to prove Lemma 5. To do this, we will need the following claim.

Claim 15. *There exists an algorithm* CalcPotential *that, given a metric space (X, d) a non-empty cluster $C \subseteq X$, and $\delta > 0$, computes a point $p \in C$ such that $\operatorname{avg}(p, C) \leq \frac{2}{|C|}\sum_{p' \in C} \operatorname{avg}(p', C)$ holds with probability $(1-\delta)$, in time $O(|C|\log(\frac{1}{\delta}))$.*

Proof. The algorithm samples $t = \lceil \log(\frac{1}{\delta}) \rceil$ points $p_1, \dots, p_t$ from C independently uniformly at random, explicitly computes $\operatorname{avg}(p_i, C)$ for each such point, and outputs the point with the lowest $\operatorname{avg}(p_i, C)$. Since computing each $\operatorname{avg}(p_i, C)$ takes only $O(|C|)$ time, the algorithm runs in time $O(|C|\log(\frac{1}{\delta}))$. So, we just need to analyze the failure probability of the algorithm.

Correctness Analysis. We need to show that the probability that all p_i simultaneously satisfy $\operatorname{avg}(p_i, C) > \frac{2}{|C|}\sum_{p' \in C} \operatorname{avg}(p', C)$ is at most δ. Since these points are sampled independently, it is enough to show that each point p_i satisfies $\operatorname{avg}(p_i, C) > \frac{2}{|C|}\sum_{p' \in C} \operatorname{avg}(p', C)$ with probability at most 1/2. Indeed, since $\mathbb{E}[\operatorname{avg}(p_i, C)] = \frac{1}{|C|}\sum_{p' \in C} \operatorname{avg}(p', C)$, where p_i is chosen uniformly from C, this probability bound is implied by Markov's inequality.

This concludes the proof of Claim 15. ☐

D.4.1 Algorithm and Analysis

To prove Lemma 5, we need to present an algorithm (Algorithm 6) that, given a metric space (X, d), a non-empty cluster $C \subseteq X$, a set $S \subseteq X$, and $\epsilon > 0$, runs in time $\tilde{O}(\frac{|C|+|S|}{\epsilon^2})$ and, for each $p \in S$, computes w.h.p a $(1+\epsilon)$-approximation of $\operatorname{avg}(p, C)$.

We will begin with the following claim about Algorithm 6.

Claim 16. *For every $p' \in S$, every $i \in \{1, \dots, t\}$, and every $p \in C$, the probability that line 15 of the algorithm will set $p_i \leftarrow p$ is exactly $\frac{d(p,p^*)+d(p',p^*)}{\sum_{p'' \in C}(d(p'',p^*)+d(p',p^*))}$.*

Proof. In the iteration of the loop in line 13 that selects p', the probability that p_i will be set to p is exactly

$$\begin{aligned}
&\frac{\operatorname{avg}(p^*, C)}{\operatorname{avg}(p^*, C) + d(p', p^*)} \cdot \Pr[p_i^{\text{weighted}} = p] + \frac{d(p', p^*)}{\operatorname{avg}(p^*, C) + d(p', p^*)} \cdot \Pr[p_i^{\text{uniform}} = p] \\
&= \frac{\operatorname{avg}(p^*, C)}{\operatorname{avg}(p^*, C) + d(p', p^*)} \cdot \frac{d(p, p^*)}{\sum_{p'' \in C} d(p'', p^*)} + \frac{d(p', p^*)}{\operatorname{avg}(p^*, C) + d(p', p^*)} \cdot \frac{1}{|C|} \\
&= \frac{\operatorname{avg}(p^*, C)}{\operatorname{avg}(p^*, C) + d(p', p^*)} \cdot \frac{d(p, p^*)}{|C| \operatorname{avg}(p^*, C)} + \frac{d(p', p^*)}{\operatorname{avg}(p^*, C) + d(p', p^*)} \cdot \frac{1}{|C|} \\
&= \frac{d(p, p^*)}{|C|(\operatorname{avg}(p^*, C) + d(p', p^*))} + \frac{d(p', p^*)}{|C|(\operatorname{avg}(p^*, C) + d(p', p^*))} \\
&= \frac{d(p, p^*) + d(p', p^*)}{\sum_{p'' \in C}(d(p'', p^*) + d(p', p^*))},
\end{aligned}$$

as needed to prove the claim. ☐

We are now ready to prove Lemma 5

Algorithm 6: CalcAverage from Lemma 5.

```
Data: $(X, d)$, $S \subseteq X$, non-empty $C \subseteq X$, $0 < \epsilon \leq 1$
Result: For each $p \in S$, as estimate $\widehat{\text{avg}}(p, C)$ of $\text{avg}(p, C)$
1  $p^* \leftarrow$ CalcCentralPoint$(C)$, $\epsilon' \leftarrow \epsilon/3$, and $t \leftarrow O(1/(\epsilon')^2)$
2  if $d(p^*, p) = 0$ for all $p \in C$ then
3      for $p' \in S$ do
4          $\widehat{\text{avg}}(p', C) \leftarrow d(p', p^*)$
5      end
6      return $\{\widehat{\text{avg}}(p', C)\}_{p' \in S}$
7  end
8  for $i = 1$ to $t$ do
9      $p_i^{\text{weighted}} \leftarrow$ sample a point $p$ from $C$ with probability proportional to $d(p, p^*)$
10     $p_i^{\text{uniform}} \leftarrow$ sample a point $p$ uniformly from $C$
11 end
12 compute $\text{avg}(p^*, C)$
13 for $p' \in S$ do
14     for $i = 1$ to $t$ do
15         $p_i \leftarrow$ sample $p_i^{\text{weighted}}$ w.p. $\frac{\text{avg}(p^*,C)}{\text{avg}(p^*,C)+d(p',p^*)}$ and $p_i^{\text{uniform}}$ w.p. $\frac{d(p',p^*)}{\text{avg}(p^*,C)+d(p',p^*)}$
16     end
17     $\widehat{\text{avg}}(p', C) \leftarrow \frac{\text{avg}(p^*,C)+d(p',p^*)}{t(1-\epsilon')} \sum_{i=1}^{t} d(p_i, p')/(d(p_i, p^*) + d(p^*, p'))$
18 end
19 return $\{\widehat{\text{avg}}(p', C)\}_{p' \in S}$
```

Proof of Lemma 5. It is straightforward to verify that Algorithm 6 runs in $O(\frac{|C|+|S|}{\epsilon^2})$ time. So, let p' be an arbitrary point in S, and our goal for the rest of the proof is to show that, w.h.p,

$$\text{avg}(p, C) \leq \widehat{\text{avg}}(p, C) \leq (1 + \epsilon)\, \text{avg}(p, C). \tag{24}$$

By Claim 15, $\text{avg}(p^*, C) \leq \frac{2}{|C|} \sum_{p'' \in C} \text{avg}(p'', C)$. So, by Lemma 1,

$$\text{avg}(p^*, C) \leq 4 \cdot \text{avg}(p', C). \tag{25}$$

For each $p \in C$, let $x_p = d(p, p')$ and let $\hat{x}_p = d(p, p^*) + d(p', p^*)$. By the triangle inequality,

$$\forall p \in C, \qquad \hat{x}_p \geq x_p \tag{26}$$

Furthermore, by the triangle inequality,

$$\sum_{p \in C} \hat{x}_p = \sum_{p \in C} (d(p, p^*) + d(p', p^*)) \leq \sum_{p \in C} (2d(p, p^*) + d(p', p)) = 2|C|\, \text{avg}(p^*, C) + |C|\, \text{avg}(p', C)$$

So, by Equation (25),

$$\sum_{p \in C} \hat{x}_p \leq 2|C|\, \text{avg}(p^*, C) + |C|\, \text{avg}(p', C) \leq 9|C|\, \text{avg}(p', C) = 9 \sum_{p \in C} x_p = O\left(\sum_{p \in C} x_p\right). \tag{27}$$

It's not difficult to see that, for our given p', the distributions of the points $p_1, \ldots, p_t$ selected by line 15 are independent. Furthermore, by Claim 16, each p_i is sampled with probability $\frac{\hat{x}_p}{\sum_{p'' \in C} \hat{x}_{p''}}$ to be each p. So, by Theorem 8, since Equation (26) and Equation (27) hold, if we correctly set the constant in the O notation at line 1 of the algorithm, then, w.h.p,

$$(1 - \epsilon') \sum_{p \in C} x_p \leq \frac{\left(\sum_{p \in C} \hat{x}_p\right)}{t} \cdot \sum_{j=1}^{t} \frac{x_{p_j}}{\hat{x}_{p_j}} \leq (1 + \epsilon') \sum_{p \in C} x_p.$$

Since $\epsilon' = \epsilon/3$ and $\epsilon \le 1$, we have $\frac{1+\epsilon'}{1-\epsilon'} \le (1+\epsilon)$, so we get that, w.h.p,

$$\sum_{p\in C} x_p \le \frac{\left(\sum_{p\in C}\hat{x}_p\right)}{t(1-\epsilon')} \cdot \sum_{j=1}^{t} \frac{x_{p_j}}{\hat{x}_{p_j}} \le (1+\epsilon)\sum_{p\in C} x_p.$$

So, by the definitions of the x_ps and $\hat{x}_p$s, w.h.p,

$$\sum_{p\in C} d(p,p') \le \frac{\sum_{p\in C}(d(p,p^*)+d(p',p^*))}{t(1-\epsilon')} \cdot \sum_{j=1}^{t} \frac{d(p_j,p')}{d(p_j,p^*)+d(p',p^*)} \le (1+\epsilon)\sum_{p\in C} d(p,p').$$

Dividing all sides of the last inequality by $|C|$, we get that, w.h.p,

$$\operatorname{avg}(p',C) \le \frac{\operatorname{avg}(p^*,C)+d(p',p^*)}{t(1-\epsilon')} \cdot \sum_{j=1}^{t} \frac{d(p_j,p')}{d(p_j,p^*)+d(p',p^*)} \le (1+\epsilon)\operatorname{avg}(p',C),$$

which implies that

$$\operatorname{avg}(p',C) \le \widehat{\operatorname{avg}}(p',C) \le (1+\epsilon)\operatorname{avg}(p',C).$$

To summarize, we saw that Algorithm 6 runs in time $\tilde{O}(\frac{|C|+|S|}{\epsilon^2})$ and we saw that for every $p' \in S$, w.h.p, the returned estimate $\widehat{\operatorname{avg}}(p',C)$ satisfies $\operatorname{avg}(p',C) \le \widehat{\operatorname{avg}}(p',C) \le (1+\epsilon)\operatorname{avg}(p',C)$. ☐

E Proof of Theorem 3

In this section, we show that if the input has a ground truth clustering $\mathcal{C}$ satisfying α-IP stability where $\alpha < 1/1000$, then it is possible to efficiently find an $O(\alpha)$-IP stable clustering.

First, we define the following quantity $\beta(\mathcal{C})$ for a clustering $\mathcal{C}$, and show how to relate this quantity to the IP stability of $\mathcal{C}$.

Definition 6. Let (X,d) be a metric space and let $C \subset X$ be a non-empty cluster. Then, we let $\beta(C)$ denote the minimum value of β that satsifies $\max_{p,p'\in C} d(p,p') \le \beta \min_{p\in C, p'\in X\setminus C} d(p,p')$. Furthermore, if $C = X$ then we let $\beta(C) \overset{\text{def}}{=} 0$. Finally, for a clustering $\mathcal{C}$ of X or of any subset of X, we let $\beta(\mathcal{C}) \overset{\text{def}}{=} \max_{C\in\mathcal{C}} \beta(C)$.

Observation 6. *For every metric space (X,d), every clustering $\mathcal{C}$ of (X,d) must be $\beta(\mathcal{C})$-IP stable*

Proof. Let $\mathcal{C}$ be a clustering in a metric space (X,d), let $p \in X$ be a point such that $C(p) \ne \{p\}$, and let $C' \in \mathcal{C}$ be a cluster other than $C(p)$. We need to show that $\operatorname{avg}(p, C(p)\setminus\{p\}) \le \beta(\mathcal{C}) \cdot \operatorname{avg}(p,C')$: By the definition of avg, for every set S, $\min_{p'\in S} d(p,p') \le \operatorname{avg}(p,S) \le \max_{p'\in S} d(p,p')$. Using this fact and the definition of $\beta(\mathcal{C})$,

$$\begin{aligned}\operatorname{avg}(p, C(p)\setminus\{p\}) \le \max_{p'\in C(p)\setminus\{p\}} d(p,p') \le \max_{p',p''\in C(p)} d(p'',p') &\le \beta(\mathcal{C}) \cdot \min_{p''\in C(p), p'\in C'} d(p'',p') \\ &\le \beta(\mathcal{C}) \cdot \min_{p'\in C'} d(p,p') \\ &\le \beta(\mathcal{C}) \cdot \operatorname{avg}(p,C'),\end{aligned}$$

as we needed. ☐

Lemma 10 (Lemma 1 in (Ahmadi et al., 2022); from Daniely et al. (2012))**.** *Let $\mathcal{C}$ be a clustering such that for every $C \ne C' \in \mathcal{C}$ and $p \in C$, $\operatorname{avg}(p,C) \le \alpha \cdot \operatorname{avg}(p,C')$, where $\alpha < 1$. Then,*

1. *For every $p \in C$, $p' \in C'$,*
$$(1-\alpha)\cdot\operatorname{avg}(p,C') \le d(p,p') \le \frac{1+\alpha^2}{1-\alpha}\cdot\operatorname{avg}(p,C'), \tag{28}$$

2. *For every $p,q \in C$ and $C' \ne C$,*
$$d(p,q) \le \frac{2\alpha}{1-\alpha}\cdot\operatorname{avg}(p,C'). \tag{29}$$

Corollary 11. *Let $\mathcal{C}$ be a clustering such that for every $C \neq C' \in \mathcal{C}$ and $p \in C$, $\text{avg}(p, C) \leq \alpha \cdot \text{avg}(p, C')$, where $\alpha < 1$. Then, for every $C \neq C' \in \mathcal{C}$,*

$$\max_{p,q \in C} d(p,q) \leq \beta \cdot \min_{p \in C, p' \in C'} d(p,p'),$$

where $\beta = \frac{2\alpha(1+\alpha^2)}{(1-\alpha)^5}$.

Proof. Let p, q denote the farthest pair of points in C. Moreover, let $x \in C, x' \in C'$ denote the closest pair of points in C and C'.

$$\begin{aligned} d(p,q) &\leq \frac{2\alpha}{1-\alpha} \text{avg}(p, C') && \triangleright \text{ By (29)} \\ &\leq \frac{2\alpha}{1-\alpha} \cdot \frac{1}{1-\alpha} \cdot d(p,x') \leq \frac{2\alpha}{(1-\alpha)^2} \cdot \frac{1+\alpha^2}{(1-\alpha)^2} \cdot \text{avg}(x', C) \leq \frac{2\alpha}{(1-\alpha)^2} \cdot \frac{1+\alpha^2}{(1-\alpha)^3} \cdot d(x,x') && \triangleright \text{ By (28)} \end{aligned}$$

□

Finally, we use the following theorem in order to find the clustering of minimal $\beta(C)$. The proof of Theorem 9 is found in Appendix E.1.

Theorem 9. *There exists a deterministic algorithm that, given a metric space (X, d) and a desired number of clusters k that admit a k-clustering $\mathcal{C}^*$ of (X, d) with $\beta(\mathcal{C}^*) < 1$, computes a k-clustering $\mathcal{C}$ of (X, d) with $\beta(\mathcal{C}) \leq \beta(\mathcal{C}^*)$.*

Now, we are ready to prove Theorem 3.

Proof of Theorem 3. Given a metric space (X, d) and a desired number of clusters k, such that there exists a k-clustering $\mathcal{C}^*$ of (X, d) that is α^*-IP stable for $\alpha^* < 0.001$. Then, by Corollary 11, $\beta(\mathcal{C}^*) \leq \frac{2\alpha^*(1+(\alpha^*)^2)}{(1-\alpha^*)^5} \leq 3\alpha^* < 1$. So, the algorithm from Theorem 9 on (X, d) and k outputs a k-clustering $\mathcal{C}$ with $\beta(\mathcal{C}) \leq \beta(\mathcal{C}^*) \leq 3\alpha^*$. Thus, by Observation 6, the clustering $\mathcal{C}$ is α-IP stable for $\alpha = \beta(\mathcal{C}) \leq 3\alpha^*$. □

E.1 Proof of Theorem 9

Algorithm 7: CreateTree

```
Data: $(X, d)$, and an MST $T$ of $(X, d)$.
Result: A binary tree $\tau$ with $n$ leaves (corresponding to the points in $X$), where each node $u$ of $\tau$
        represents a subset $f(u)$ of $X$ such that $f(u) = f(\text{left-child}(u)) \cup f(\text{right-child}(u))$.
1  $root \leftarrow$ new node
2  $f(root) \leftarrow X$
3  if $|X| = 1$ then
4      left-child($root$) $\leftarrow$ null
5      right-child($root$) $\leftarrow$ null
6      return $root$
7  end
8  $e_{\max} \leftarrow$ the edge of $T$ with maximum length
9  $C_1, C_2 \leftarrow$ the two connected components of $T \setminus \{e_{\max}\}$
10 $(X_1, d), (X_2, d)$ the restrictions of $(X, d)$ to $C_1$ and $C_2$
11 $T_1, T_2 \leftarrow$ the restrictions of $T$ to $C_1$ and $C_2$
12 left-child($root$) $\leftarrow$ CreateTree($(X_1, d), T_1$)
13 right-child($root$) $\leftarrow$ CreateTree($(X_2, d), T_2$)
14 return $root$
```

Definition 7. Let (X, d) be a metric space and let τ be the output of Algorithm 7 on this metric space. Then, we say that a clustering $\mathcal{C}$ of X is *induced* by the tree τ if, for every cluster $C \in \mathcal{C}$, there exists a node u of τ with $f(u) = C$.

Claim 17. *For every metric space (X, d) and every clustering $\mathcal{C}$ of (X, d) with $\beta(\mathcal{C}) < 1$, $\mathcal{C}$ must be induced by the tree τ resulting from Algorithm 7 on (X, d) and a minimum spanning tree of it.*

Lemma 11. *There exists a deterministic polynomial time algorithm that, given a metric space (X, d), a desired number of clusters k, and a tree τ returned by Algorithm 7 on (X, d), computes a k-clustering $\mathcal{C}$ with $\beta(\mathcal{C}) = \min_{\mathcal{C}' \in \mathcal{S}_\tau} \beta(\mathcal{C}')$, where $\mathcal{S}_\tau$ denote the set of k-clusterings induced by τ.*

Proof of Theorem 9. Let (X, d) be a metric space that admit a k-clustering $\mathcal{C}^*$ with $\beta(\mathcal{C}^*) < 1$. To find a k-clustering $\mathcal{C}$ with $\beta(\mathcal{C}) \leq \beta(\mathcal{C}^*)$, first we compute a minimum spanning tree T of (X, d), then run Algorithm 7 on (X, d) and T to produce a tree τ, and finally run the algorithm from Lemma 11 on the metric space (X, d) and the tree τ with parameter k.

By Claim 17, the clustering $\mathcal{C}^*$ must be induced by the tree τ. Therefore, the algorithm from Lemma 11 must return a clustering $\mathcal{C}$ with $\beta(\mathcal{C}) \leq \beta(\mathcal{C}^*)$, as we need. Since computing a minimum spanning tree of (X, d) can be done in polynomial time, and since Algorithm 7 and the algorithm from Lemma 11 both run in polynomial time, this suffices to prove Theorem 9. □

E.1.1 Proof of Claim 17

In order to prove Claim 17, we first need to prove the following claim.

Claim 18. *Let τ be the output of Algorithm 7 on (X, d) and a minimum spanning tree of (X, d) as inputs, where (X, d) admits an k-clustering $\mathcal{C}$ with $\beta(\mathcal{C}) < 1$. Then, every node in τ represents either a subset of clusters in $\mathcal{C}$, or a subset of a cluster in $\mathcal{C}$.*

Proof. We show that for every node $u \in \tau$, if there exists $p \in C, q \in C'$ where $C \neq C'$, such that $\{p, q\} \subseteq f(u)$, then $f(u)$ contains all points in $C \cup C'$.

Suppose this property does not hold for all nodes in τ. consider a node $v \in \tau$ closest to the root that does not satisfy this property. Note that the root itself trivially satisfies the property. Let v_p denote the parent node of v in τ. Without loss of generality, suppose that v does not contain all points in C and its intersection with C' is non-empty. Since v is the closest node to the root that does not satisfy the described property, v_p contains all points in $C \cup C'$. Let T_{v_p} be the restriction of the MST T on the points contained in v_p.

First, we prove that $T_{v_p}[C]$, the induced subgraph of T_{v_p} on C is a connected component. Otherwise, $T_{v_p}[C]$ has at least two connected components C_1, C_2 such that both $T_{v_p}[C_1]$ and $T_{v_p}[C_2]$ are trees and they both have distinct edges leaving C_1 and C_2 in T_{v_p}. However, it contradicts the minimality of T_{v_p}, and consequently the minimality of T. This follows, because by the condition $\beta(\mathcal{C}) < 1$, all edges within C are strictly smaller than any edge with one end-point in C and one end-point outside of C. So, if we add an arbitrary edge between C_1, C_2 to T, then we can remove one of the edges leaving C; hence, the cost of T after this modification strictly decrease which is a contradiction.

Now, if by removing an edge in T_v the points in C are split into at least two non-empty sets, then the edge has to be between two points in C. However, this is not possible because T_{v_p} is a tree spanning both C and C' and by the condition $\beta(\mathcal{C}) < 1$ any edge between C and C' is strictly large than any edge corresponding to two points in C. So, all nodes in τ satisfy the desired property. □

We are now ready to prove Claim 17

Proof of Claim 17. The proof is by contradiction. Lets assume that $\mathcal{C}$ is not induced by the tree τ. Then, there must exist a cluster $C \in \mathcal{C}$ such that no node v in τ represents the set $C \subseteq X$. Let v^C be the highest node in τ, i.e., closest to the root, that represents a set $f(v^C) \subseteq C$. Note that such a node exists because, for every point $p \in X$, there exists a leaf node in τ that represents the set $\{p\}$. By the choice of C, $f(v^C) \neq C$, so $f(v^C)$ is strictly contained in C,

$$f(v^C) \subset C, \tag{30}$$

which means that $f(v^C) \neq X$, and thus v^C has a parent v^C_{parent} and a sibling v^C_{sibling}. By the choice of v^C, since v^C_{parent} is closer to the root than v^C, there must exist some cluster $C' \in \mathcal{C}$ other than C such that $f(v^C_{\text{parent}}) \cap C' \neq$

$\emptyset$. By the design of Algorithm 7 and the choice of v^C, $f(v^C)$ is non-empty and $f(v^C) \subseteq f(v^C_{\text{parent}}) \cap C$, which means that $f(v^C_{\text{parent}}) \cap C \neq \emptyset$. By Claim 18, since $f(v^C_{\text{parent}}) \cap C' \neq \emptyset$ and $f(v^C_{\text{parent}}) \cap C \neq \emptyset$, we must have

$$C \cup C' \subseteq f(v^C_{\text{parent}}). \tag{31}$$

Since $f(v^C)$ is strictly contained in C (Equation (30)) and since C' is non-empty, there must be at least one point in C and at least one point in C' that are not in $f(v^C)$. By Equation (31) and by the fact that $f(v^C_{\text{parent}}) = f(v^C) \cup f(v^C_{\text{sibling}})$, these two points must both belong to $f(v^C_{\text{sibling}})$; hence, by Claim 18, $C \cup C' \subseteq f(v^C_{\text{sibling}})$. So, $f(v^C) \subseteq C \subseteq f(v^C_{\text{sibling}})$, which contradicts the structure of τ.

Thus, the clustering $\mathcal{C}$ must be induced by τ. This concludes the proof of Claim 17. $\square$

E.1.2 Proof of Lemma 11

Definition 8. Given a metric space (X, d) and the tree τ from Algorithm 7. For every node u of τ, we say that a clustering $\mathcal{C}_u$ of $f(u)$ is induced by the subtree of τ rooted at u if, for every cluster $C \in \mathcal{C}_u$, there exists a node v in the subtree of τ rooted at u such that $C = f(v)$. We note that $\beta(\mathcal{C}_u)$ still refers to the quantity $\max_{C \in \mathcal{C}_u} \beta(C)$, where each $\beta(C)$ is defined with respect to the whole space X.

Proof of Lemma 11. We follow a dynamic programming approach. For each tuple (u, k') where u is a node in τ and $1 \leq k' \leq k$, $DP(u, k')$ denotes the k'-clustering $\mathcal{C}_u$ of $f(u)$ induced by the subtree of τ rooted at u minimizing $\beta(\mathcal{C}_u)$, or Null if such a k'-clustering does not exist. Note that for any node u, $DP(u, 1) = \{f(u)\}$, i.e., there is a trivial 1-clustering of $f(u)$ which is a single cluster containing $f(u)$. Furthermore, for every leaf node ℓ and every $k' > 1$, $DP(\ell, k') = \text{Null}$ because there is no way to partition $f(\ell)$ into $k' > 1$ disjoint non-empty clusters, as $|f(\ell)| = 1$. For every node $u \in \tau$, let u_R and u_L respectively denote the right and left child of u. Then, the update rule in the dynamic programming is as follows:

- $DP(u, 1) = \{f(u)\}$, where $f(u)$ is the set of points contained in u.
- $DP(u, k' > 1) = \arg\min_{\mathcal{C}_{u,k'} \in \text{Candidates}_{u,k'}} \beta(\mathcal{C}_{u,k'})$ where

  $$\text{Candidates}_{u,k'} \overset{\text{def}}{=} \{DP(u_R, i) \cup DP(u_L, k' - i) \mid i \in [k' - 1] \text{ s.t. } DP(u_R, i) \text{ and } DP(u_L, k' - i) \text{ are not Null}\},$$

  or $DP(u, k' > 1) = \text{Null}$ if $\text{Candidates}_{u,k'}$ is empty.

The dynamic programming has $|\tau| k = O(nk)$ cells and updating each of them requires at most $O(kn^2)$ time, the total runtime of the algorithm is $O(n^3 k^2)$. Furthermore, by definition, the value of $DP(\text{root}(\tau), k)$ is exactly the desired clustering that we need to compute.

Regarding the correctness of the algorithm, it follows by an inductive argument and the definition of clustering induced by τ together with Claim 17. $\square$

F Proof of Theorem 4

In this section, we prove Theorem 4. To do this, we first present a general framework in Appendix F.1 for extending the techniques of Section 3 to other types of IP stability. Then, throughout most of this section, we show how to apply this framework in order to construct an algorithm that, given (X, d) and a desired number of clusters k, produces an $(O(1), \sqrt{\text{median}})$-IP stable k-clustering. To finish of, we use the following theorem, concluding that such a clustering must also be $(O(1), \text{median})$-IP stable.

So, proving Theorem 4 requires the following theorem and lemma. We will now show how to prove Theorem 4 using these, and show the proof of the lemma. The proof of Theorem 10 is in Appendix F.7.

Theorem 10. *There exists a deterministic polynomial-time algorithm that, given a metric space (X, d) and a desired number of clusters $2 \leq k \leq |X|$, computes an $(O(1), \sqrt{\text{median}})$-IP stable k-clustering of (X, d), where $\sqrt{\text{median}}$ denotes the square root of the median function from Theorem 4.*

Lemma 12. *Given a metric space (X, d), every $(\alpha, \sqrt{\text{median}})$-IP stable cluster of (X, d) must also be $(\alpha^2, \text{median})$-IP stable.*

Proof of Theorem 4 using Theorem 10 and Lemma 12. We use the algorithm from Theorem 10. By Lemma 12, the clustering that this algorithm returns must be $(O(1), \text{median})$-IP stable. Therefore, this algorithm satisfies all the properties promised in Theorem 4. □

Proof of Lemma 12. Given a metric space (X, d), if a clustering $\mathcal{C}$ is $(\alpha, \sqrt{\text{median}})$-IP stable, then, for every point p with $C(p) \neq \{p\}$ and every cluster $C' \in \mathcal{C} \setminus \{C\}$, $\sqrt{\text{median}(p, C(p) \setminus \{p\})} \leq \alpha \cdot \sqrt{\text{median}(p, C')}$. So,

$$\text{median}(p, C(p) \setminus \{p\}) \leq \alpha^2 \cdot \text{median}(p, C')$$

which is exactly the condition required for $\mathcal{C}$ to be $(\alpha^2, \text{median})$-IP stable. □

F.1 Framework for General f-IP Stability

In this subsection, we show how the techniques from Section 3 can be extended to a framework for constructing algorithms that compute f-IP stable clusterings for other functions f.

Theorem 11. *For every metric space (X, d), let $f : X \times 2^X \to \mathbb{R}_{\geq 0}$ and $\Phi_f : (2^X \setminus \{\emptyset\}) \to \mathbb{R}_{\geq 0}$ and $\alpha > 0$. Furthermore, suppose the following properties hold:*

1. *For every non-empty subset $S \subset X$ and every point $p \in X \setminus S$,*
$$f(p, S) \leq \Phi_f(S \cup \{p\}) - \Phi_f(S) \leq \alpha \cdot f(p, S).$$
2. *There exists a deterministic polynomial-time algorithm that computes $f(p, S)$, given any $S \subset X$, and $p \in X \setminus S$.*
3. *There exists a deterministic polynomial-time algorithm that computes a $\text{poly}(n)$-approximation of $\Phi_f(C)$, given any $C \subset X$.*
4. *There exists a deterministic polynomial-time algorithm* Split$_f$ *that, given a clustering $\mathcal{C}$ of X, finds a cluster $C^* \in \mathcal{C}$ and a partition (C_1^*, C_2^*) of C^* such that $\Phi(C_1^*) + \Phi(C_2^*) \leq \Phi(C^*) - \frac{\Phi(\mathcal{C})}{\text{poly}(n)}$.*
5. *For every two disjoint non-empty clusters $C, C' \subseteq X$, and every $p \in C$,*
$$\Phi_f(C \cup C') \leq \Phi_f(C) + \Phi_f(C') + \text{poly}(n) \cdot (f(p, C \setminus \{p\}) + f(p, C'))$$
6. *There exists a deterministic polynomial time algorithm* Initialize$_f$ *that, given a metric space (X, d) and a desired number of clusters k, finds a k-clustering $\mathcal{C}$ that $\text{poly}(n)$-approximately minimizes the potential function $\Phi_f(\mathcal{C})$; more precisely, $\Phi_f(\mathcal{C}) \leq \text{poly}(n) \cdot \Phi_f(\mathcal{C}^*)$, where $\mathcal{C}^*$ is a k-clustering minimizing Φ_f, and $\Phi_f(\mathcal{C}) \overset{def}{=} \sum_{C \in \mathcal{C}} \Phi_f(C)$.*

Then, for every constant $c > 1$, there exists a deterministic polynomial-time algorithm that, given a desired number of clusters k, computes a $(c\alpha, f)$-IP stable k-clustering of (X, d).

Since the proof of Theorem 11 is analogous to the arguments in Section 3, we only provide the following sketch.

Proof Sketch for Theorem 11. The algorithm is analogous to Algorithm 2. It begins by initializing a clustering using the algorithm from property 6 in Theorem 11. Then, as long as the current clustering is not $(c\alpha, f)$-IP stable (which can be checked using the algorithm from property 2), it picks a point p and cluster that violate the constraint of $(c\alpha, f)$-IP stability, and produces an estimate of $\Phi_f(\mathcal{C})$ using the algorithm from property 3 of Theorem 11. If the estimate of $\Phi_f(\mathcal{C})$, the value of $f(p, C \setminus \{p\}) + f(p, C')$, and properties 4 and 5 of Theorem 11 are together enough to guarantee that a merge-and-split step would decrease the overall potential of the clustering by a multiplicative $(1 - \frac{1}{\text{poly}(n)})$ factor, then it executes a merge-and-split step, merging clustering $C(p)$ and C', then using the algorithm from property 4 to split some cluster C^*. Otherwise, it must be that $f(p, C \setminus \{p\}) + f(p, C') \geq \frac{\Phi_f(\mathcal{C})}{\text{poly}(n)}$, so, since p and C' violate $(c\alpha, f)$-IP stability, it must be that $f(p, C \setminus \{p\}) \geq \frac{1}{2}(f(p, C \setminus \{p\}) + f(p, C')) \geq \frac{\Phi_f(\mathcal{C})}{\text{poly}(n)}$, so, by property 1 of Theorem 11, swapping the point p from cluster $C(p)$ to cluster C' must reduce the overall potential by

$$f(p, C \setminus \{p\}) - \alpha f(p, C') > f(p, C \setminus \{p\}) - \frac{\alpha}{c\alpha} f(p, C \setminus \{p\}) = \Omega(f(p, C \setminus \{p\})) \geq \frac{\Phi_f(\mathcal{C})}{\text{poly}(n)},$$

and this is what the algorithm does.

In summary, using all the properties from Theorem 11, the algorithm is able to initialize a k-clustering $\mathcal{C}$ in polynomial time such that $\Phi_f(\mathcal{C})$ is a poly(n)-approximation of $\Phi_f(\mathcal{C}^*)$, where $\mathcal{C}^*$ is a minimizer of Φ_f, and then continue in iterations as long as the current maintained clustering is not $(c\alpha, f)$-IP stable, where each iteration runs in polynomial time and reduces the potential Φ_f of the maintained clustering $\mathcal{C}$ by a multiplicative factor $< (1 - \frac{1}{\text{poly}(n)})$. Thus, since the potential of the maintained clustering can never go below $\Phi_f(\mathcal{C}^*)$, the number of iterations made by the algorithm is bounded by poly(n), so the overall runtime of the algorithm is polynomial. $\square$

F.2 Potential Function and Preliminaries for $\sqrt{\text{median}}$-IP

In this section, we introduce the potential function $\Phi_{\sqrt{\text{median}}}$. Just as the potential function discussed in Section 2.1 played a crucial role in Section 3, this new function will be integral to the framework presented in Appendix F.1. Furthermore, we prove some preliminary observations for analyzing this potential function.

Definition 9 (Potential function for median-IP)**.** Given a metric space (X, d), let $G = (X, E, w)$ be the complete clique graph on X, together with edge-lengths $w(p_1, p_2) \overset{\text{def}}{=} \sqrt{d(p_1, p_2)}$ for each edge (p_1, p_2) of the clique. Then, we define the potential function $\Phi_{\sqrt{\text{median}}} : (2^X \setminus \{\emptyset\}) \to \mathbb{R}_{\geq 0}$ such that, for every non-empty subset $C \subseteq X$,

$$\Phi_{\sqrt{\text{median}}}(C) \overset{\text{def}}{=} \begin{cases} \frac{1}{2-\sqrt{2}} \cdot \max_{\text{TSP tour } \Pi \text{ of } G[C]} w(\Pi) & \text{if } |C| \geq 2 \\ 0 & \text{if } |C| = 1 \end{cases}$$

where $G[C]$ is the subgraph of G induced on the set C of vertices.[3]

To use this potential function in the framework of Appendix F.1, we need to prove the following theorem. The proof of the theorem is in Appendix F.3.

Theorem 12. *For every metric space (X, d), every non-empty set $S \subset X$ and every point $p \in X \setminus S$, the potential function from Definition 9 satisfies*

$$\sqrt{\text{median}(p, S)} \leq \Phi_{\sqrt{\text{median}}}(S \cup \{p\}) - \Phi_{\sqrt{\text{median}}}(S) \leq O(1) \cdot \sqrt{\text{median}(p, S)},$$

where median(p, S) *is defined as in Theorem 4.*

In order to analyze the potential function defined in this section, the following observations will be useful.

Observation 7. *Let $G = (V, E)$ be a complete clique graph with at least two vertices, and let Π be a TSP tour of G. If a partition (A, B) of V satisfies $|B| > |A|$, then there must be an edge $(p, p') \in E(\Pi)$ such that $p, p' \in B$.*

Observation 8. *Given a metric space (X, d) and the graph G as defined in Definition 9, for every three points $p, p', p'' \in X$,*

$$w(p', p'') \leq w(p', p) + w(p, p''),$$

and thus

$$w(p', p'') - w(p, p'') \leq w(p, p').$$

Observation 9. *For every metric space (X, d) and every non-empty subset $C \subseteq X$,*

$$\sqrt{\text{diameter}(C)} \leq \Phi_{\sqrt{\text{median}}}(C) \leq O(|C| \cdot \sqrt{\text{diameter}(C)}).$$

Corollary 12. *For every metric space (X, d) and every non-empty subset $\mathcal{C}$,*

$$\max_{C \in \mathcal{C}} \sqrt{\text{diameter}(C)} \leq \sum_{C \in \mathcal{C}} \Phi_{\sqrt{\text{median}}}(C) \leq O(n \cdot \max_{C \in \mathcal{C}} \sqrt{\text{diameter}(C)}).$$

[3] A TSP tour Π of $G[C]$ is a Hamiltonian cycle in $G[C]$, and its length is $w(\Pi) = \sum_{e \in E(\Pi)} w(e) = \sum_{(p_1, p_2) \in E(\Pi)\}} \sqrt{d(p_1, p_2)}$. If $|C| = 2$, i.e. if $C = \{p_1, p_2\}$ for some $p_1, p_2 \in X$, then we consider Π to be taking the edge (p_1, p_2) twice, which means that $w(\Pi) = 2w(p_1, p_2) = 2\sqrt{d(p_1, p_2)}$.

F.2.1 Proofs of Observations

In this section, we prove Observation 7 and Observation 8.

Proof of Observation 7. If the clique graph $G = (V, E)$ has exactly two vertices, then the TSP tour Π must take the edge between these two vertices, and the condition $|B| > |A|$ implies that $B = V$, which means that this edge has both its endpoints in B. So, for the rest of the proof, we can assume that $|V| \neq 2$. Then, by the condition $|V| \geq 2$, we have $|V| > 2$. Therefore, each vertex $v \in B$ must have at least two incident edges of $E(\Pi)$. So, since $|B| > |A|$ and since each vertex of A can have at most two incident edges of $E(\Pi)$,

$$\sum_{v \in B} \deg_{E(\Pi)}(v) \geq 2|B| > 2|A| \geq \sum_{v \in A} \deg_{E(\Pi)}(v).$$

Since (A, B) is a partition of V, the above inequality implies that there must exist some edge of $E(\Pi)$ whose endpoints are both in B. $\square$

Proof of Observation 8. By the definition of the lengths w and by the triangle inequality

$$w(p', p'') = \sqrt{d(p', p'')} \leq \sqrt{d(p', p) + d(p, p'')}$$

so, by the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, and by again using the definitions of the lengths w,

$$w(p', p'') \leq \sqrt{d(p', p) + d(p, p'')} \leq \sqrt{d(p', p)} + \sqrt{d(p, p'')} = w(p', p) + w(p, p'').$$

$\square$

Proof of Observation 9. Firstly, if $|C| = 1$, then $\Phi_{\sqrt{\text{median}}}(C)$ and $\sqrt{\text{diameter}(C)}$ are both zero, so the inequalities hold. Thus, for the rest of the proof, we assume $|C| \geq 2$. So, let $p, p' \in C$ be two point in C such that $d(p, p') = \text{diameter}(C)$. Then, there exists a TSP tour of C that uses the edge (p, p'), which means that

$$w(p, p') \leq \max_{\text{TSP tour } \Pi \text{ of } G[C]} w(\Pi) \leq \Phi_{\sqrt{\text{median}}}(C).$$

So, since we picked the point p, p' such that $d(p, p') = \text{diameter}(C)$,

$$\sqrt{\text{diameter}(C)} = \sqrt{d(p, p')} = w(p, p') \leq \Phi_{\sqrt{\text{median}}}(C).$$

Furthermore, for every TSP tour Π of $G[C]$, since Π contains exactly $|C|$ edges, and the length of each edge $e \in E(\Pi)$ is $w(e) = \sqrt{d(e)} \leq \sqrt{\text{diameter}(C)}$, we get that every TSP tour Π of $G[C]$ has $w(\Pi) \leq |C| \cdot \sqrt{\text{diameter}(C)}$, which exactly means that

$$\Phi_{\sqrt{\text{median}}}(C) \leq \frac{1}{2 - \sqrt{2}} \cdot |C| \cdot \sqrt{\text{diameter}(C)} = O(|C| \cdot \sqrt{\text{diameter}(C)}).$$

Thus, we proved both sides of the desired inequality from Observation 9, which concludes the proof of the observation. $\square$

Proof of Corollary 12. By Observation 9, each cluster $C \in \mathcal{C}$ satisfies

$$\Phi_{\sqrt{\text{median}}}(C) \leq O(|C| \cdot \sqrt{\text{diameter}(C')}) \leq O(|C| \cdot \max_{C' \in \mathcal{C}} \sqrt{\text{diameter}(C')}).$$

So,

$$\begin{aligned} \sum_{C \in \mathcal{C}} \Phi_{\sqrt{\text{median}}}(C) \leq \sum_{C \in \mathcal{C}} O(|C| \cdot \max_{C' \in \mathcal{C}} \sqrt{\text{diameter}(C')}) &\leq \left(\sum_{C \in \mathcal{C}} |C| \right) \cdot O(\max_{C' \in \mathcal{C}} \sqrt{\text{diameter}(C')}) \\ &= O(n \cdot \max_{C' \in \mathcal{C}} \sqrt{\text{diameter}(C')}) \\ &= O(n \cdot \max_{C \in \mathcal{C}} \sqrt{\text{diameter}(C)}) \end{aligned}$$

Furthermore, letting $C_{\max}$ be the cluster with largest diameter in $\mathcal{C}$, then using Observation 9,

$$\max_{C \in \mathcal{C}} \sqrt{\text{diameter}(C)} = \sqrt{\text{diameter}(C_{\max})} \leq \Phi_{\sqrt{\text{median}}}(C_{\max}) \leq \sum_{C \in \mathcal{C}} \Phi_{\sqrt{\text{median}}}(C).$$

Thus, we proved both sides of the desired inequality from Corollary 12, concluding the proof of the corollary. $\square$

F.3 Proof of Theorem 12

We being by proving the following claim.

Claim 19. *Given a metric space (X, d), a set $S \subset X$ with $|S| \geq 2$, a point $p \in X \setminus S$, and a TSP tour $\Pi_{S\cup\{p\}}$ of the graph $G[S \cup \{p\}]$ as defined in Definition 9 and Theorem 12. If there exists a point $p' \in S$ such that $d(p, p') \leq \operatorname{median}(p, S)$ and the edge (p', p) belongs to the TSP tour $\Pi_{S\cup\{p\}}$, then there exists a TSP tour Π_S of $G[S]$ such that $w(\Pi_S) \geq w(\Pi_{S\cup\{p\}}) - O(1) \cdot \sqrt{\operatorname{median}(p, S)}$.*

Proof of Claim 19. Since $|S| \geq 2$, there must be some point $p'' \in S$ other than p' such that $(p, p'') \in E(\Pi_{S\cup\{p\}})$. To construct the tour Π_S to be like $\Pi_{S\cup\{p\}}$ except the two edges (p', p) and (p, p'') are replaced by the single edge (p', p''), thus skipping over the point p. Therefore,

$$w(\Pi_S) - w(\Pi_{S\cup\{p\}}) = w(p', p'') - w(p', p) - w(p, p'').$$

So, by Observation 8,

$$w(\Pi_S) - w(\Pi_{S\cup\{p\}}) \leq -2w(p', p) = -2\sqrt{d(p', p)}.$$

Therefore, since we are given that $d(p, p') \leq \operatorname{median}(p, S)$,

$$w(\Pi_S) \geq w(\Pi_{S\cup\{p\}}) - 2\sqrt{d(p', p)} \geq w(\Pi_{S\cup\{p\}}) - 2\sqrt{\operatorname{median}(p, S)},$$

as we needed to prove. □

We are now ready to prove Theorem 12.

Proof of Theorem 12. Let (X, d), S, and p be as in the theorem, and let G be defined as in Definition 9. We will first deal with the case of $|S| = 1$, and then separately prove each of the two inequalities in Theorem 12 when $|S| \geq 2$.

Case $|S| = 1$. In this case there exists some $p' \in X$ such that $S = \{p'\}$, so $\Phi_{\sqrt{\text{median}}}(S) = 0$ and $\Phi_{\sqrt{\text{median}}}(S \cup \{p\}) = \frac{1}{2-\sqrt{2}} \cdot 2\sqrt{d(p, p')}$, while $\operatorname{median}(p, S) = d(p, p')$. This immediately implies that $\sqrt{\operatorname{median}(p, S)} \leq \Phi_{\sqrt{\text{median}}}(S \cup \{p\}) - \Phi_{\sqrt{\text{median}}}(S) \leq O(1) \cdot \sqrt{\operatorname{median}(p, S)}$, as we needed.

Left-Hand Side Inequality. In this paragraph, our goal is to prove the left-hand side inequality from Theorem 12, under the additional assumption $|S| \geq 2$. In other words, we need to show that $\Phi_{\sqrt{\text{median}}}(S) + \sqrt{\operatorname{median}(p, S)} \leq \Phi_{\sqrt{\text{median}}}(S \cup \{p\})$. So, let Π_S be the maximum-length TSP tour in $G[S]$, which means that $\Phi_{\sqrt{\text{median}}}(S) = \frac{1}{2-\sqrt{2}} \cdot w(\Pi_S)$. We need to construct a TSP tour $\Pi_{S\cup\{p\}}$ of $G[S \cup \{p\}]$ with $\frac{1}{2-\sqrt{2}} \cdot w(\Pi_{S\cup\{p\}}) \geq \frac{1}{2-\sqrt{2}} \cdot w(\Pi_S) + \sqrt{\operatorname{median}(p, S)}$. Let $A \stackrel{\text{def}}{=} \{p' \in S \mid d(p', p) < \operatorname{median}(p, S)\}$ and let $B \stackrel{\text{def}}{=} \{p' \in S \mid d(p', p) \geq \operatorname{median}(p, S)\}$. By the definition of $\operatorname{median}(p, S)$, $|B| > |A|$. (See Theorem 4.) Since (A, B) is a partition of S with $|S| \geq 2$ and $|B| > |A|$, by Observation 7, there must be some edge $(p_1, p_2) \in E(\Pi_S)$ such that $p_1, p_2 \in B$. We build the TSP tour $\Pi_{S\cup\{p\}}$ in $G[S \cup \{p\}]$ by starting with Π_S, then removing the edge (p_1, p_2) and instead adding the edges (p_1, p) and (p, p_2). It's not hard to see that

$$w(\Pi_{S\cup\{p\}}) - w(\Pi_S) = w(p_1, p) + w(p, p_2) - w(p_1, p_2) = \sqrt{d(p_1, p)} + \sqrt{d(p, p_2)} - \sqrt{d(p_1, p_2)}.$$

Thus, by the triangle inequality,

$$w(\Pi_{S\cup\{p\}}) - w(\Pi_S) \geq \sqrt{d(p_1, p)} + \sqrt{d(p, p_2)} - \sqrt{d(p_1, p) + d(p, p_2)}.$$

So, using the inequality $\sqrt{a} + \sqrt{b} - \sqrt{a+b} - (2 - \sqrt{2})\sqrt{\min\{a, b\}} \geq 0$,[4]

$$w(\Pi_{S\cup\{p\}}) - w(\Pi_S) \geq (2 - \sqrt{2})\sqrt{\min\{d(p_1, p), d(p, p_2)\}}.$$

[4]It's very easy to see that this inequality holds for all $a, b \geq 0$. The desired values of a and b can always be achieved by first starting with setting $a = b$ and then increasing only one of a,b. Setting $a = b$ causes exact equality to hold, then increasing one of them can only increase the left-hand side expression, since $\min\{a, b\}$ does not change.

Since $p_1, p_2 \in B$, they satisfy $\min\{d(p_1,p), d(p,p_2)\} \geq \mathrm{median}(p,S)$, so

$$w(\Pi_{S\cup\{p\}}) - w(\Pi_S) \geq (2-\sqrt{2})\sqrt{\mathrm{median}(p,S)},$$

which exactly gives the inequality

$$\frac{1}{2-\sqrt{2}} \cdot w(\Pi_{S\cup\{p\}}) \geq \frac{1}{2-\sqrt{2}} \cdot w(\Pi_S) + \sqrt{\mathrm{median}(p,S)},$$

as we needed. This concludes the proof of $\sqrt{\mathrm{median}(p,S)} \leq \Phi_{\sqrt{\mathrm{median}}}(S\cup\{p\}) - \Phi_{\sqrt{\mathrm{median}}}(S)$.

Right-Hand Side Inequality In this paragraph, our goal is to prove the right-hand side inequality from Theorem 12, under the additional assumption $|S| \geq 2$. In other words, we need to show that $\Phi_{\sqrt{\mathrm{median}}}(S \cup \{p\}) - O(1) \cdot \sqrt{\mathrm{median}(p,S)} \leq \Phi_{\sqrt{\mathrm{median}}}(S)$. So, let $\Pi_{S\cup\{p\}}$ be the maximum-length TSP tour in $G[S\cup\{p\}]$, which means that $\Phi_{\sqrt{\mathrm{median}}}(S\cup\{p\}) = \frac{1}{2-\sqrt{2}} \cdot w(\Pi_{S\cup\{p\}})$. We need to show that there exists a TSP tour Π_S of $G[S]$ with $\frac{1}{2-\sqrt{2}} \cdot w(\Pi_S) \geq \frac{1}{2-\sqrt{2}} \cdot w(\Pi_{S\cup\{p\}}) - O(1) \cdot \sqrt{\mathrm{median}(p,S)}$, which is an equivalent condition to $w(\Pi_S) \geq w(\Pi_{S\cup\{p\}}) - O(1) \cdot \sqrt{\mathrm{median}(p,S)}$. To do this, we will construct a TSP tour $\Pi'_{S\cup\{p\}}$ of $G[S\cup\{p\}]$ that has the property required by Claim 19 and satisfies $w(\Pi'_{S\cup\{p\}}) \geq w(\Pi_{S\cup\{p\}}) - O(1) \cdot \sqrt{\mathrm{median}(p,S)}$.

Firstly, if the tour $\Pi_{S\cup\{p\}}$ itself has the property required by Claim 19, then the claim implies the existence of a tour Π_S as we need, and then we are done. So, for the rest of the proof, we assume that the tour $\Pi_{S\cup\{p\}}$ doesn't satisfy the property from Claim 19, meaning that there is no point $p' \in S$ such that both $(p,p') \in E(\Pi_{S\cup\{p\}})$ and $d(p,p') \leq \mathrm{median}(p,S)$ hold. Since $|S| \geq 2$, this implies that there are two distinct points $p_1, p_2 \in S$ such that $(p_1,p), (p,p_2) \in E(\Pi_{S\cup\{p\}})$ and $d(p,p_1), d(p,p_2) > \mathrm{median}(p,S)$. Let $A \overset{\mathrm{def}}{=} \{p' \in S \mid d(p',p) \leq \mathrm{median}(p,S)\}$ and let $B \overset{\mathrm{def}}{=} \{p' \in S \mid d(p',p) > \mathrm{median}(p,S)\}$. (Notice that unlike in the previous section of the proof, the inequality in the definition of B is strict and the one in the definition of A is not.) Therefore, by the definition of $\mathrm{median}(p,S)$, $|A| \geq |B|$, which means that $|A \cup \{p\}| > |B|$. Thus, by Observation 7, there must be some edge $(p'_1, p') \in \Pi_{S\cup\{p\}}$ such that $p'_1, p' \in A \cup \{p\}$. Since $|S \cup \{p\}| \geq 3$, there must be some point p'_2 other than p'_1 such that $(p', p'_2) \in E(\Pi_{S\cup\{p\}})$. Furthermore, by the assumption $d(p,p_1), d(p,p_2) > \mathrm{median}(p,S)$, we have $p_1, p_2 \in B$. Therefore, both incident edges of p in $E(\Pi_{S\cup\{p\}})$ lead from it to a point B, while each of p'_1, p', p'_2 has an incident edge of $E(\Pi_{S\cup\{p\}})$ leading from them to a point outside of B, so $p \notin \{p'_1, p', p'_2\}$. This means that the edges $(p_1,p),(p,p_2)$ are both distinct from the edges $(p'_1,p'),(p',p'_2)$. Now, we construct the TSP tour $\Pi'_{S\cup\{p\}}$ of $G[S\cup\{p\}]$ to be like $\Pi_{S\cup\{p\}}$ except that the edges $(p_1,p),(p,p_2),(p'_1,p'),(p',p'_2)$ are replaced by $(p_1,p'),(p',p_2),(p'_1,p),(p,p'_2)$. Then, the tour $\Pi'_{S\cup\{p\}}$ satisfies the property from Claim 19 because $(p'_1,p) \in E(\Pi'_{S\cup\{p\}})$ and because $p'_1 \in A\cup\{p\}$ means that $d(p,p'_1) \leq \mathrm{median}(p,S)$. Furthermore,

$$\begin{aligned} w(\Pi'_{S\cup\{p\}}) - w(\Pi_{S\cup\{p\}}) &= w(p_1,p') + w(p',p_2) + w(p'_1,p) + w(p,p'_2) - w(p_1,p) - w(p,p_2) - w(p'_1,p') - w(p',p'_2) \\ &= (w(p_1,p') - w(p_1,p)) + (w(p',p_2) - w(p,p_2)) + (w(p'_1,p) - w(p'_1,p')) \\ &\quad + (w(p,p'_2) - w(p,p'_2)). \end{aligned}$$

Thus, by Observation 8,

$$w(\Pi'_{S\cup\{p\}}) - w(\Pi_{S\cup\{p\}}) \geq (-w(p',p)) + (-w(p',p)) + (-w(p',p)) + (-w(p',p)) = -4w(p',p) = -4\sqrt{d(p',p)}.$$

So, since $p' \in A \cup \{p\}$,

$$w(\Pi'_{S\cup\{p\}}) \geq w(\Pi_{S\cup\{p\}}) - 4\sqrt{d(p',p)} \geq w(\Pi_{S\cup\{p\}}) - 4\sqrt{\mathrm{median}(p,S)} = w(\Pi_{S\cup\{p\}}) - O(\sqrt{\mathrm{median}(p,S)}).$$

In summary, we proved that there exists a TSP tour $\Pi'_{S\cup\{p\}}$ of $G[S\cup\{p\}]$ that satisfies the conditions of Claim 19 and also satisfies $w(\Pi'_{S\cup\{p\}}) \geq w(\Pi_{S\cup\{p\}}) - O(\sqrt{\mathrm{median}(p,S)})$. So, by Claim 19, there exists a TSP tour Π_S of $G[S]$ with

$$w(\Pi_S) \geq w(\Pi'_{S\cup\{p\}}) - O(\sqrt{\mathrm{median}(p,S)}) \geq w(\Pi_{S\cup\{p\}}) - O(\sqrt{\mathrm{median}(p,S)}).$$

This implies that the TSP tour $w(\Pi_S)$ satisfies

$$\frac{1}{2-\sqrt{2}}w(\Pi_S) \geq \frac{1}{2-\sqrt{2}}w(\Pi'_{S\cup\{p\}}) - O(\sqrt{\text{median}(p,S)}) = \Phi_{\sqrt{\text{median}}}(S\cup\{p\}) - O(\sqrt{\text{median}(p,S)}),$$

and thus that $\Phi_{\sqrt{\text{median}}}(S) \geq \Phi_{\sqrt{\text{median}}}(S\cup\{p\}) - O(\sqrt{\text{median}(p,S)})$, as we needed to prove.

This concludes the proof of Theorem 12. □

F.4 Split Procedure for $\sqrt{\text{median}}$-IP

In this section, we prove the following lemma.

Lemma 13. *There exists a deterministic polynomial-time algorithm that, given a metric space (X,d) and a clustering $\mathcal{C}$ of X, finds a cluster $C^* \in \mathcal{C}$ and a partition (C_1^*, C_2^*) of C^* with $\Phi_{\sqrt{\text{median}}}(C_1^*) + \Phi_{\sqrt{\text{median}}}(C_1^*) \leq \Phi_{\sqrt{\text{median}}}(C^*) - \left(\sum_{C\in\mathcal{C}} \Phi_{\sqrt{\text{median}}}(C)\right)/\text{poly}(n)$.*

To do this, we first need the following claim.

Claim 20. *Let (X,d) be a metric space, let $C \subseteq X$, and let $p,p' \in C$. Then,*

$$\text{median}(p, C\setminus\{p\}) + \text{median}(p', C\setminus\{p'\}) \geq d(p,p')$$

Proof of Claim 20. Let $A \overset{\text{def}}{=} \{p'' \in C \mid d(p'',p) \leq \text{median}(p, C\setminus\{p\})\}$ and $B \overset{\text{def}}{=} \{p'' \in C \mid d(p'',p) > \text{median}(p, C\setminus\{p\})\}$ and $A' \overset{\text{def}}{=} \{p'' \in C' \mid d(p'',p) \leq \text{median}(p, C\setminus\{p'\})\}$ and $B' \overset{\text{def}}{=} \{p'' \in C' \mid d(p'',p) > \text{median}(p, C\setminus\{p'\})\}$. Then, $|A| \geq |B|$ and $|A'| \geq |B'|$, which means that $|A\cup\{p\}| > |B|$ and $|A'\cup\{p'\}| > B'$. Since $(A\cup\{p\}, B)$ and $(A'\cup\{p'\}, B')$ are both partitions of C, we get that $|A\cup\{p\}| > |C|/2$ and $|A'\cup\{p'\}| > |C|/2$ and thus that there must be some point $p'' \in (A\cup\{p\}) \cap (A'\cup\{p'\})$. Therefore, $d(p,p'') \leq \text{median}(p, C\setminus\{p\})$ and $d(p',p'') \leq \text{median}(p', C\setminus\{p'\})$. So, by the triangle inequality,

$$d(p,p') \leq d(p,p'') + d(p',p'') \leq \text{median}(p, C\setminus\{p\}) + \text{median}(p', C\setminus\{p'\}).$$

as we needed to prove. □

We are now ready to prove the lemma. The algorithm from the lemma is Algorithm 8.

Algorithm 8: SPLIT: The cluster splitting procedure from Lemma 13.

```
    Data: $(X,d)$ where $|X| = n$, and a clustering $\mathcal{C}$.
    Result: a cluster $C^* \in \mathcal{C}$ and a partition of $C^*$ satisfying the condition described in Lemma 13.
1   $C^* \leftarrow \arg\max_{C\in\mathcal{C}} \max_{p,p'\in C} d(p,p')$
2   $p, p' \leftarrow \arg\max_{p,p'\in C^*} d(p,p')$
3   $p'' \leftarrow \arg\max_{p''\in\{p,p'\}} \text{median}(p'', C^*\setminus\{p''\})$
4   return $C^*$ and the partition $(C^*\setminus\{p''\}, \{p''\})$
```

Proof or Lemma 13. Firstly, By Definition 9, $\Phi_{\sqrt{\text{median}}}(\{p''\}) = 0$. Secondly, by Theorem 12,

$$\Phi_{\sqrt{\text{median}}}(C^*\setminus\{p''\}) \leq \Phi_{\sqrt{\text{median}}}(C^*) - \sqrt{\text{median}(p'', C^*\setminus\{p''\})}.$$

Thirdly, by Claim 20 and by the choice of p'',

$$\text{median}(p'', C^*\setminus\{p''\}) \geq \frac{1}{2}(\text{median}(p, C^*\setminus\{p\}) + \text{median}(p', C^*\setminus\{p'\})) \geq \frac{1}{2}d(p,p').$$

Putting these three together, then using the way that p and p' were chosen,

$$\begin{aligned}\Phi_{\sqrt{\text{median}}}(C^* \setminus \{p''\}) + \Phi_{\sqrt{\text{median}}}(\{p''\}) &\le \Phi_{\sqrt{\text{median}}}(C^*) - \sqrt{\text{median}(p'', C^* \setminus \{p''\})} \\ &\le \Phi_{\sqrt{\text{median}}}(C^*) - \sqrt{\frac{1}{2}d(p,p')} \\ &= \Phi_{\sqrt{\text{median}}}(C^*) - \max_{C' \in \mathcal{C}} \max_{p,p' \in C'} \sqrt{\frac{1}{2}d(p,p')} \\ &= \Phi_{\sqrt{\text{median}}}(C^*) - \max_{C' \in \mathcal{C}} \max_{p,p' \in C'} \frac{1}{\sqrt{2}} w(p,p')\end{aligned} \tag{32}$$

Furthermore, by Corollary 12,

$$\sum_{C \in \mathcal{C}} \Phi_{\sqrt{\text{median}}}(C) \le O(n \cdot \max_{C' \in \mathcal{C}} \sqrt{\text{median}(C')}) = O(n \cdot \max_{C' \in \mathcal{C}} \max_{p,p' \in C'} w(p,p')).$$

Combining the last inequality and Equation (32),

$$\Phi_{\sqrt{\text{median}}}(C^* \setminus \{p''\}) + \Phi_{\sqrt{\text{median}}}(\{p''\}) \le \Phi_{\sqrt{\text{median}}}(C^*) - \frac{1}{O(n)} \left(\sum_{C \in \mathcal{C}} \Phi_{\sqrt{\text{median}}}(C) \right),$$

as we needed to prove.

This concludes the proof of Lemma 13. ☐

F.5 Bounding Potential Increase of Cluster Merges for $\sqrt{\text{median}}$-IP

The goal of this section is to prove the following lemma.

Lemma 14. *For every metric space (X, d) with $|X| = n$, every two disjoint non-empty clusters $C, C' \subseteq X$, and every point $p \in X$, if $\Phi_{\sqrt{\text{median}}}$ is the potential function from Definition 9, then*

$$\Phi_{\sqrt{\text{median}}}(C \cup C') \le \Phi_{\sqrt{\text{median}}}(C) + \Phi_{\sqrt{\text{median}}}(C') + \text{poly}(n) \cdot (\sqrt{\text{median}(p, C)} + \sqrt{\text{median}(p, C')}).$$

To prove the above lemma, we will need the following claims.

Claim 21. *Given a metric space (X, d) with $|X| = n$, a subset $S \subseteq X$ with $|S| \ge 2$, a point $p \in X$, and a radius $r > 0$, and a subset $A \subseteq S$ such that every point $p' \in A$ satisfies $w(p', p) \le r$. Then, every TSP tour Π_S of $G[S]$, as defined in Definition 9, satisfies*

$$w(\Pi_S) \le 2 \left(\sum_{p' \in S \setminus A} w(p', p) \right) + \text{poly}(n) \cdot r$$

Proof. By Observation 8,

$$\begin{aligned}w(\Pi_S) = \sum_{(p',p'') \in E(\Pi_S)} w(p', p'') \le \sum_{(p',p'') \in E(\Pi_S)} (w(p', p) + w(p, p'')) &= 2 \sum_{p' \in S} w(p, p') \\ &= 2 \left(\sum_{p' \in S \setminus A} w(p, p') \right) + 2 \left(\sum_{p' \in A} w(p, p') \right) \\ &\le 2 \left(\sum_{p' \in S \setminus A} w(p, p') \right) + 2 \left(\sum_{p' \in A} r \right) \\ &= 2 \left(\sum_{p' \in S \setminus A} w(p, p') \right) + \text{poly}(n) \cdot r\end{aligned}$$

☐

Claim 22. *For every metric space (X,d) with $|X| = n$, every subset $S \subseteq X$ with $|S| \geq 2$, and every a point $p \in X$, there exists a TSP tour Π_S of $G[S]$, as defined in Definition 9, such that*

$$w(\Pi_S) \geq 2\left(\sum_{p' \in S \mid d(p',p) > \mathrm{median}(p,S)} w(p',p)\right) - \mathrm{poly}(n) \cdot \sqrt{\mathrm{median}(p,S)}$$

Proof. Let $A \overset{\text{def}}{=} \{p' \in S \mid d(p',p) \leq \mathrm{median}(p,S)\}$ and $B \overset{\text{def}}{=} \{p' \in S \mid d(p',p) > \mathrm{median}(p,S)\}$. Then, $|A| \geq |B|$, so let $A' \subseteq A$ be some subset of A with $|A'| = |B|$. We construct the tour Π_S by starting with a path that alternates between A' and B until visiting all vertices of these sets, and then extending that path to visit the vertices of $A \setminus A'$, and finally coming back to the starting vertex. Then, for every $p' \in B$, the two edges of Π_S that are incident to p' must both have their other endpoint in A, so, by Observation 8 and by the definition of A, they must each have length at least $w(p',p) - \sqrt{\mathrm{median}(p,S)}$. Therefore,

$$\begin{aligned}\sum_{e \in \Pi_S} w(e) \geq 2\sum_{p' \in B}(w(p',p) - \sqrt{\mathrm{median}(p,S)}) &= 2\sum_{p' \in S \mid d(p',p) > \mathrm{median}(p,S)} (w(p',p) - \sqrt{\mathrm{median}(p,S)}) \\ &= 2\left(\sum_{p' \in S \mid d(p',p) > \mathrm{median}(p,S)} w(p',p)\right) - \mathrm{poly}(n) \cdot \sqrt{\mathrm{median}(p,S)}\end{aligned}$$

□

Now, we are ready to prove Lemma 14.

Proof of Lemma 14. Firstly, if $|C| = 1$ or $|C'| = 1$, then the lemma follows from Theorem 12. So, for the rest of the proof, we assume that $|C| \geq 2$ and $|C'| \geq 2$. Let $A \overset{\text{def}}{=} \{p' \in C \mid d(p',p) \leq \mathrm{median}(p,C)\}$ and $B \overset{\text{def}}{=} \{p' \in C \mid d(p',p) > \mathrm{median}(p,C)\}$ and $A' \overset{\text{def}}{=} \{p' \in C' \mid d(p',p) \leq \mathrm{median}(p,C')\}$ and $B' \overset{\text{def}}{=} \{p' \in C' \mid d(p',p) > \mathrm{median}(p,C')\}$. By Claim 21, since all points $p' \in A \cup A'$ satisfy $w(p',p) \leq \sqrt{\mathrm{median}(p,C)} + \sqrt{\mathrm{median}(p,C')}$,

$$\Phi_{\sqrt{\mathrm{median}}}(C \cup C') \leq \frac{1}{2-\sqrt{2}} \cdot 2\left(\sum_{p' \in B \cup B'} w(p',p)\right) + \mathrm{poly}(n) \cdot (\sqrt{\mathrm{median}(p,C)} + \sqrt{\mathrm{median}(p,C')}).$$

Furthermore, by Claim 22,

$$\Phi_{\sqrt{\mathrm{median}}}(C) \geq \frac{1}{2-\sqrt{2}} \cdot 2\left(\sum_{p' \in B} w(p',p)\right) - \mathrm{poly}(n) \cdot \sqrt{\mathrm{median}(p,C)}$$

and

$$\Phi_{\sqrt{\mathrm{median}}}(C') \geq \frac{1}{2-\sqrt{2}} \cdot 2\left(\sum_{p' \in B'} w(p',p)\right) - \mathrm{poly}(n) \cdot \sqrt{\mathrm{median}(p,C')}.$$

Subtracting the last two inequalities from the first one,

$$\Phi_{\sqrt{\mathrm{median}}}(C \cup C') - \Phi_{\sqrt{\mathrm{median}}}(C) - \Phi_{\sqrt{\mathrm{median}}}(C') \leq \mathrm{poly}(n) \cdot (\sqrt{\mathrm{median}(p,C)} + \sqrt{\mathrm{median}(p,C')}),$$

as we needed to prove. □

F.6 Initialization for $\sqrt{\mathrm{median}}$-IP

In this section, we show that initializing using the greedy k-centers algorithm also works for the potential function $\Phi_{\sqrt{\mathrm{median}}}$ from Definition 9. In other words, we prove the following lemma, which is an analogue to Lemma 4.

Lemma 15. *Given a metric space (X,d) with n points and a k-clustering $\mathcal{C}$. If $\mathcal{C}$ is a 2-approximate optimal solution to the k-centers problem, then it* $\mathrm{poly}(n)$*-approximately minimizes the potential function $\Phi_{\sqrt{\mathrm{median}}}$; more precisely, $\Phi_{\sqrt{\mathrm{median}}}(\mathcal{C}) \leq \mathrm{poly}(n) \cdot \Phi_{\sqrt{\mathrm{median}}}(\mathcal{C}^*)$, where $\mathcal{C}^*$ is a k-clustering minimizing $\Phi_{\sqrt{\mathrm{median}}}$.*

Proof of Lemma 15. Since $\mathcal{C}$ is a 2-approximate k-centers solution, and since radius is a 2-approximation of diameter,

$$\max_{C\in\mathcal{C}} \text{diameter}(C) \leq 4 \cdot \max_{C\in\mathcal{C}^*} diamter(C).$$

So, by Corollary 12,

$$\Phi_{\sqrt{\text{median}}}(\mathcal{C}) \leq O(n \cdot \max_{C\in\mathcal{C}} diamter(C)) \leq O(n \cdot \max_{C\in\mathcal{C}^*} diamter(C)) \leq O(n \cdot \Phi_{\sqrt{\text{median}}}(\mathcal{C}^*)),$$

as we needed to prove. □

F.7 Putting it All Together

In this section, we prove Theorem 10 using the framework of Appendix F.1.

Proof of Theorem 10. In order to prove the theorem, we just need to show that for every metric space (X, d), the function $\sqrt{\text{median}} : X \times 2^X \to \mathbb{R}_{\geq 0}$ and the potential function $\Phi_{\sqrt{\text{median}}}$ from Definition 9 satisfy all the conditions of Theorem 11 for $\alpha = O(1)$. We will now do this by going over each of the conditions:

1. The first condition from Theorem 11 is satisfied by Theorem 12.
2. It is clear that $\sqrt{\text{median}}(p, S)$ can be computed in polynomial time given (X, d), $S \subseteq X$, and p, so the second condition holds.
3. Using Corollary 12, it is clear that a poly(n) approximation of $\Phi_{\sqrt{\text{median}}}(\mathcal{C})$ can be computed in polynomial time, by simply computing $\max_{C\in\mathcal{C}} \text{diameter}(C)$. So, the third condition holds. We note that an $O(1)$-approximation can be achieved via an approximate maximum matching algorithm, but we will not prove this.
4. By Lemma 13, the fourth condition holds.
5. By Lemma 14, the fifth condition holds.
6. By Lemma 15 and the classic 2-approximate k-centers algorithm, the sixth condition holds.

In summary, we saw that all the conditions of Theorem 11 are satisfied for $f = \sqrt{\text{median}}$ with $\alpha = O(1)$. So, by Theorem 11, there exists a deterministic polynomial time algorithm that, given a metric space (X, d) and a desired number of clusters k, computes a $(O(1), \sqrt{\text{median}})$-IP stable k-clustering of (X, d). This concludes the proof of Theorem 10. □

G Proof of Theorem 5

In this section, we prove Theorem 5, showing that every metric space (X, d) and number of clusters $2 \leq k \leq |X|$ admit a $(1, \max)$-IP stable k-clustering, where $\max(p, S) \overset{\text{def}}{=} \max_{p'\in S} d(p, p')$. To do this, we will prove that the natural local search algorithm for the problem of finding such a clustering (Algorithm 9) must terminate. (Though the running time of the algorithm is not polynomial.)

Algorithm 9: MaxIP-LS: Local Search For Max-IP.

```
    Data: $(X, d)$, $2 \leq k \leq n = |X|$.
    Result: A $(1, \max)$-IP stable $k$-clustering of $(X, d)$.
1   initialize an arbitrary $k$-clustering $\mathcal{C}$ of $(X, d)$
2   while exists $p$ and $C'$ s.t. $C(p) \neq \{p\}$ and $\max(p, C(p) \setminus \{p\}) > \max(p, C')$ do
3       move $p$ to cluster $C'$
4   end
5   return the current clustering;
```

First, we need the following definition and claim. We will use these to prove Theorem 5, and then we will prove Claim 23.

Definition 10 (Potential function for max-IP)**.** Consider the complete clique graph $G = (X, E)$ on X where each edge $(p, p') \in E$ has length $w(p, p') = d(p, p')$. Label the edges of G so that $w(e_1) \geq w(e_2) \geq \ldots \geq w(e_{|E|})$. For every clustering $\mathcal{C}$ of X, let $s_{\mathcal{C}} \in \{0,1\}^{|E|} = \{0,1\}^{\binom{|X|}{2}}$ be the string whose ith character is a 1 if both endpoints of e_i are in the same cluster of $\mathcal{C}$ and is 0 otherwise.

Claim 23. *At each step of Algorithm 9, the string $s_{\mathcal{C}}$ corresponding to the maintained clustering $\mathcal{C}$, as in Definition 10, goes down in lexicographic order.*

Proof of Theorem 5 using Claim 23. Consider an execution of Algorithm 9 on (X, d) with parameter k. Claim 23 implies that the clustering $\mathcal{C}$ maintained by the algorithm never returns to a state that it was in before. Therefore, since there are a finite number of possible clusterings of (X, d), Algorithm 9 must terminate in finite time. When the algorithm terminates, the condition of the while loop in line 2 must not be satisfied, which exactly means that the current clustering $\mathcal{C}$ maintained by the algorithm is $(1, \max)$-IP stable. Furthermore, it is not hard to see that the number of clusters in $\mathcal{C}$ always remains k. Thus, the algorithm terminates and outputs a $(1, \max)$-IP stable k-clustering of (X, d). Therefore, such a clustering exists. □

To conclude this section, we will now prove Claim 23.

Proof of Claim 23. Given a specific step of Algorithm 9, let $\mathcal{C}$ be the state of the clustering at the beginning of the step, let p and C' be the point and cluster selected in this step, let C be the cluster containing p at the beginning of the step, and let $\mathcal{C}'$ be the state of the clustering at the end of the step. Let

$$I_{\text{reduced}} \overset{\text{def}}{=} \{i \mid \text{the } i\text{th character of } s_{\mathcal{C}} \text{ is 1 and the } i\text{th character of } s_{\mathcal{C}'} \text{ is } 0\}$$

and

$$I_{\text{increased}} \overset{\text{def}}{=} \{i \mid \text{the } i\text{th character of } s_{\mathcal{C}} \text{ is 0 and the } i\text{th character of } s_{\mathcal{C}'} \text{ is } 1\}.$$

In order to show that $s_{\mathcal{C}'}$ is smaller than $s_{\mathcal{C}}$ in lexicographic order, we need to show that I_{reduced} is non-empty and that $\min\{i \in I_{\text{reduced}}\} < \min\{i \in I_{\text{increased}}\}$. Since the difference between $\mathcal{C}$ and $\mathcal{C}'$ is that p was moved from C to C',

$$I_{\text{reduced}} = \{i \mid e_i \text{ has one endpoint in } C \setminus \{p\} \text{ and the other endpoint is } p\}$$

and

$$I_{\text{increased}} = \{i \mid e_i \text{ has one endpoint in } C' \text{ and the other endpoint is } p\}.$$

Since $C \neq \{p\}$, the set I_{reduced} cannot be empty. Furthermore, since the edges are ordered in non-increasing order of their length, and since $\max(p, C \setminus \{p\}) > \max(p, C')$, we must have $\min\{i \in I_{\text{reduced}}\} < \min\{i \in I_{\text{increased}}\}$, as we needed.

In summary, we showed that for each step of Algorithm 9, there is at least one i such that the step turns the ith character of the string $s_{\mathcal{C}}$ from 1 to 0, and that the minimum i for which this happens is smaller than the minimum i for which the step turns the ith character of the string $s_{\mathcal{C}}$ from 0 to 1. Thus, each step of the algorithm reduces the string $s_{\mathcal{C}}$ in lexicographic order. This concludes the proof of Claim 23 □

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes. Due to the space constraint, several proofs are moved to the appendix]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes. Due to the space constraints, several of them are moved to the appendix]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not Applicable]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]