On Algorithms for Large-Scale Graph and Clustering Problems

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Chris Schwiegelshohn

Dortmund

2017

Tag der mündlichen Prüfung:	25. August 2017
Dekan:	Prof. Dr. Gernot Fink
Gutachter:	Prof. Dr. Christian Sohler
	Prof. Dr. Stefano Leonardi

Zusammenfassung

Gegenstand dieser Arbeit sind algorithmische Methoden der modernen Datenanalyse. Dabei werden vorwiegend zwei übergeordnete Themen behandelt: *Datenstromalgorithmen* mit Kompressionseigenschaften und Approximationsalgorithmen für *Clusteringverfahren*. Datenstromalgorithmen verarbeiten einen Datensatz sequentiell und haben das Ziel, Eigenschaften des Datensatzes (approximativ) zu bestimmen, ohne dabei den gesamten Datensatz abzuspeichern. Unter Clustering versteht man die Partitionierung eines Datensatzes in verschiedene Gruppen.

Das erste dargestellte Problem betrifft Matching in Graphen. Hier besteht der Datensatz aus einer Folge von Einfüge- und Löschoperationen von Kanten. Die Aufgabe besteht darin, die Größe des so genannten Maximum Matchings so genau wie möglich zu bestimmen. Es wird ein Algorithmus vorgestellt, der, unter der Annahme, dass das Matching höchstens die Größe k hat, die exakte Größe bestimmt und dabei k^2 Speichereinheiten benötigt. Dieser Algorithmus lässt sich weiterhin verwenden um eine konstante Approximation der Matchinggröße in planaren Graphen zu bestimmen. Des Weiteren werden untere Schranken für den benötigten Speicherplatz bestimmt und eine Reduktion von gewichtetem Matching zu ungewichteten Matching durchgeführt.

Anschließend werden Datenstromalgorithmen für die Nachbarschaftssuche betrachtet, wobei die Aufgabe darin besteht, für n gegebene Mengen die Paare mit hoher Ähnlichkeit in nahezu Linearzeit zu finden. Dabei ist der *Jaccard Index* $\frac{|A \cap B|}{|A \cup B|}$ das Ähnlichkeitsmaß für zwei Mengen A und B. In der Arbeit wird eine Datenstruktur beschrieben, die dies erstmalig in dynamischen Datenströmen mit geringem Speicherplatzverbrauch leistet. Dabei werden Zufallszahlen mit nur 2-facher Unabhängigkeit verwendet, was eine sehr effiziente Implementierung ermöglicht.

Das dritte Problem befindet sich an der Schnittstelle zwischen den beiden Themen dieser Arbeit und betrifft das k-center Clustering Problem in Datenströmen mit einem Zeitfenster. Die Aufgabe besteht darin k Zentren zu finden, so dass die maximale Distanz unter allen Punkten zu dem jeweils nächsten Zentrum minimiert wird. Ergebnis sind ein 6-Approximationalgorithmus für ein beliebiges k und ein optimaler 4-Approximationsalgorithmus für $k = 2$. Die entwickelten Techniken lassen sich ebenfalls auf das Durchmesserproblem anwenden und ermöglichen für dieses Problem einen optimalen Algorithmus.

Danach werden Clusteringprobleme bezüglich der Jaccard Distanz analysiert. Dabei sind wieder eine Menge N von Teilmengen aus einer Grundgesamtheit U sind und die Aufgabe

besteht darin eine Teilmenge C zu finden, die $\max_{X \in N} 1 - \frac{|X \cap C|}{|X \cup C|}$ minimiert. Es wird gezeigt, dass zwar eine exakte Lösung des Problems NP-schwer ist, es aber gleichzeitig eine PTAS gibt.

Abschließend wird die weit verbreitete lokale Suchheuristik für k-median und k-means Clustering untersucht. Obwohl es Im Allgemeinen schwer ist, diese Probleme exakt oder auch nur approximativ zu lösen gelten sie in der Praxis als relativ gut handhabbargelten diese Probleme jedoch als handhabbar, was andeutet, dass die Härteresultate auf pathologischen Eingaben beruhen. Auf Grund dieser Diskrepanz gab es in der Vergangenheit praxisrelevante Datensätze zu charakterisieren. Für drei der wichtigsten Charakterisierungen wird das Verhalten einer lokalen Suchheuristik untersucht mit dem Ergebnis, dass die lokale Suchheuristik in diesen Fällen optimale oder fast optimale Cluster ermittelt. Diese Ergebnisse erklären, warum die lokale Suchheuristik in der Praxis gute Ergebnisse liefert und deuten ebenfalls daraufhin, dass die Charakterisierungen tatsächlich auf praxisrelevante Datensätze zutreffen.

Summary

In this thesis, we investigate algorithmic approaches to modern data analysis tasks. Broadly speaking, we addresses two major topics: Compression algorithms implementable in *streaming* models and efficient approximation algorithms for *clustering* problems. Streaming algorithms read a data set in a sequential fashion and (approximately) answer queries while using significantly less space than the entire input. At first, we focus on matching in graphs. Based on a sequence of edge insertions and deletions to an adjacency matrix we want to determine an estimate of the maximum matching size at any given time in the stream. We show that if the matching size is limited to k then we can determine the exact matching size using roughly k^2 space. As a by-product, we also use this algorithm to obtain a constant factor estimate to the matching size in planar graphs in sublinear space. Further, we provide lower bounds for small approximation ratios and a weighted to unweighted matching reduction.

Next, we investigate nearest-neighbor searching problems in dynamic streams. In this problem, we aim to determine pairs of high similarity among n item sets in linear or almost linear time. Our similarity measure is the Jaccard index, defined as $\frac{|A \cap B|}{|A \cup B|}$ for two item sets A and B. We design a data structure that allows us to filter out low-similarity pairs while storing significantly less space than the entire input set. This data structure is the first one that can process dynamic streams and requires only 2-wise independent random variables, enabling an efficient implementation.

Bridging both topics of this thesis, we then investigate clustering in streaming models. In the k-center problem, we search for k points that minimize the maximum distance of any input point to its nearest center. For sliding window streams, we obtain a 6-approximation for every value of k and an optimal 4-approximation for $k = 2$. Our techniques also lead to an optimal algorithm for the diameter problem.

Moving on, we study the 1-center problem with Jaccard distance. For an arbitrary collection N of subsets of a ground set U, we want to find a subset C that minimizes $\max_{X \in N} 1 - \frac{|X \cap C|}{|X \cup C|}$. We show that the problem is NP-hard, but admits a PTAS.

Lastly, we investigate the popular Local Search heuristic for k-median and k-means clustering. While these problems are known to be difficult to approximate in the worst-case, folklore wisdom regards clustering to be feasible in practice. Thus, previous research has attempted to formally capture practical instances. We identify the three most important definitions of the last few years and analyze the performance of Local Search for each of them. We prove that Local Search finds optimal or near optimal clusterings for all of these definitions, which on the one hand shows why Local Search works well in practice and on the other hand it justifies that these definitions do indeed capture practically relevant instances.

Contents

1 Introduction and Overview

In the digital age, data has become arguably the most important natural resource. The value of data arises less as a singular occurrence, but through global trends that only reveal themselves once sufficiently many data points accumulate. Therein lies the major challenge of modern data analysis: On the one hand, the larger the data set, the more we can expect it to contain the global trends. On the other hand, the size of the data set limits the applicability of tools we might be willing or able to employ to detect those trends. This thesis is motivated by the desire to find and analyze algorithms sitting in the sweet spot between information sufficiency and scalability.

Information Compression A central paradigm to handle large-scale data sets is to compress them such that the relevant information is mostly retained. Thereafter, we may use an algorithm of choice. Compression, as understood in data analysis, is different from the lossless or near lossless compression for files. Instead, we allow some loss in terms of accuracy for more manageable data sets. Often, a reduced accuracy is acceptable, especially if the global trends are the result of statistic behavior, as the information in the entire data set itself will only be a fuzzy approximation to a ground truth. The quality of a compression algorithm is then measured as a trade-off between accuracy and space.

But sometimes a terrific accuracy-space trade-off is not enough. In high throughput applications, the data can arrive with such velocity and volume that accessing each item more than once or accessing the items in some ordered fashion is infeasible. For instance, the data measured in experiments at the Large Hadron Collider in Cern arrives in magnitudes of Gigabyte per second. One way to treat such problems theoretically is the *streaming model*, where we read the data set, one item at a time in an online fashion and at the end produce a succinct summary that approximates the behavior of the original data set.

Information Extraction The second central part of data analysis is to find a computationally efficient method that reveals the information we are interested in. This has been one of the central topics of algorithm design even before the advent of big data, but the work is still ongoing with many open questions. Indeed the need to process modern data sets, that may be quite large even after compression, sets the requisites of many algorithms. Ideally, the algorithm is simple, fast in practice (though not necessarily in theory) and can be counted on to solve the problem in some quantifiable way. Of course, whether a given algorithm satisfies

these requirements can vary wildly depending on the perspective. Simplicity is always in the eye of the beholder, fast running times in practice can only be agreed upon after an extended period of utilization and barring a claim to optimality, there is no way to determine whether an algorithm solves the problem in a way that is universally agreed upon. Nevertheless, theoretical algorithm designers have begun to address the first two points by analyzing algorithms that have already prevailed in practice and exploring to what degree these algorithms have a satisfactory worst-case behavior on any instance for a given problem.

Problems Studied in this Thesis In order to describe our contribution, we informally introduce the problems appearing in the following chapters. We view the data either as organized in form of a graph or as points in Euclidean space. A graph is a tuple $G(V, E)$, where V are nodes and $E \subseteq V \times V$ are edges. One of the most widely studied graph problems is *matching*, where we aim to find the largest set of edges M such that no two edges in M share a common node. Matching is not only popular due to its theoretical appeal, it also features heavily in online advertising and as a subroutine for many other problems. Another other graph-problem is *nearest neighbor searching*. Here, we aim to quickly determine nodes that are highly similar, where in our case the measure of similarity is the Jaccard index.

The other major analysis topic is *clustering*, which roughly corresponds to partitioning the data into k sets called clusters such that the items in the same cluster are similar and the items in different clusters are dissimilar. There are many ways to define cluster criteria. In this thesis we will focus on center based clustering objectives where the cost of clustering is a function of the distances between each point and its nearest cluster.

1.1 Streaming Algorithms

In the basic streaming model [254], we view the data as a high-dimensional vector X and the stream consists of a sequence of updates to X. We are tasked with solving some problem given X as input, with stringent space constraints. The problems may consist of maintaining vector statistics such as the number of distinct entries of X, to more complex problems such as matching and clustering. Without storing the entire vector, we usually cannot solve the problem in question optimally or exactly and thus we resort to some degree of approximation. The achievable quality of the approximation depends on the complexity of the problem, but also whether we can make any further assumptions on the updates. For instance, *insertion-only streams*, where every entry of X is modified only once, are often far easier to handle than *turnstile streams*, where each entry of X can have an arbitrary number of additive updates.

Often, the entries of X will have further semantics, such as when X is the adjacency matrix of a graph $G(V, E)$. Most classic graph problems do not admit $o(|V|)$ streaming algorithms, which gives rise to the semi-streaming model introduced by Feigenbaum et al. [141]. Here, we are allowed to use $O(|V| \text{ polylog } |V|)$ space. The arguably most studied problem in graph

streaming is matching [244], where the accuracy-space trade-offs for matching in various update models are well if not completely understood. Thus recent research has begun to study the problem of approximating the matching size in streams [163].

Another popular assumption is that the entries of X contain the coordinates of points in Euclidean space and the points in their entirety are added one by one. Compressing such a sequence of points with respect to clustering objectives has been a popular topic of research even before the theoretical groundwork on streaming [289]. Subsequently, much theoretical work has been done to improve streaming algorithms for various center-based clustering objectives.

1.2 Approximation Algorithms

Many clustering tasks are NP-hard [164, 171], meaning that barring a major algorithmic breakthrough, it is unlikely that we can compute optimal clusterings in general. Nevertheless, clustering is a widespread and successful tool to organize, aggregate, and understand data. Here, approximation algorithms help us understand this discrepancy, and give us a tool for finding clusterings even when computing the optimum is hard. The approximation ratio is the ratio between the objective value of the solution computed by an algorithm and the objective value of the optimum solution.

For many NP-hard clustering objectives, algorithms exist that compute solutions with a $(1+\varepsilon)$-approximation ratio [25], where $\varepsilon > 0$ is an adjustable parameter of the algorithm. In these cases, while some difficult "core" of the problem seems to exist, we can find a clustering that is, in some sense, close to the optimum in reasonable time.

But sometimes computing a solution with a $(1+\varepsilon)$-approximation ratio is also NP-hard. In many cases, such hardness results do not reflect practical experiences. Often, the hard instances are pathological and do not have meaningful clusterings [257]. In these cases, we might try to characterize meaningful instances and design algorithms with good approximation ratios only on those instances.

1.3 Organization and Results

Chapter 2 We define the basic notions and mathematical tools we will use throughout this thesis. We also formally define the studied problems and models and give a state-of-the-art overview on the literature for each problem. Previously published results with a specific bearing on one of the results presented in this thesis are included at the beginning of the corresponding chapter.

Chapter 3 In this chapter, we study the problem of approximating the matching size in dynamic graph streams. We show that if the matching size is promised to be at most k, there

exists a $\tilde{O}(k^2)$ space algorithm that exactly determines the matching size. With this algorithm as a subroutine, we obtain the first constant approximation for estimating matchings in planar graphs when the edges arrive in a dynamic data stream. In addition, we show that any algorithm maintaining a $(1 + O(\varepsilon))$ approximation to the matching size in an insertion-only data stream must use at least $\Omega(n^{1-\varepsilon})$ space.

Lastly, we present improved results on estimating weighted matchings. Roughly speaking, when given an α-approximate unweighted matching estimator, we can use that algorithm to obtain a 2α-approximated weighted matching estimator with only a small increase in space.

Chapter 4 We study the nearest neighbor searching in dynamic data streams using the Jaccard index. Given two sets A, B, the Jaccard index is $\frac{|A\cap B|}{|A\cup B|}$. Our goal is to design a data structure maintainable in dynamic streams such that given n sets, we can find all item pairs whose Jaccard index exceeds a given threshold in near linear time. As a by-product, we show that the corresponding Jaccard-distance $1 - \frac{|A\cap B|}{|A\cup B|}$ can be $(1 + \varepsilon)$-approximated in dynamic streams. The analysis requires only 2-wise independent random variables, which allows us a very efficient implementation, which we further confirm via a short experimental evaluation.

Chapter 5 Sliding window streams consider only the most recent N inserted items. Given a sequence of points lying in some metric space, we study the problem of estimating the diameter of the most recent N points and the k-center clustering problem. In the former case, we give an algorithm storing a constant number of points with an approximation ratio of 3 and show that any algorithm with an approximation ratio smaller than 3 has to store $\text{poly}(N)$ points.

For 2-center, we give an algorithm with an approximation ratio of 4 and for k-center, we give an algorithm with an approximation ratio of 6. We also show that any algorithm with an approximation ratio smaller than 4 for 2-center has to store $\text{poly}(N)$ points.

Chapter 6 In this chapter we study the Jaccard-center problem, where we are given a collection N of subsets of some ground set U and aim to determine a set C such that $\max_{X\in N} 1 - \frac{|X\cap C|}{|X\cup C|}$ is minimized. We show that the problem is NP-hard, but can be $(1+\varepsilon)$-approximated in polynomial time for any fixed $\varepsilon > 0$.

To describe the last result, we abuse notation and denote by C and X the binary vectors corresponding to the sets. We show that Jaccard-center is fixed parameter tractable in the Hamming norm of the vector $X - C$, which arises naturally in this type of problem.

Chapter 7 In this chapter, we aim to understand the behavior of the Local Search heuristic on well-behaved clustering instances. There exists many attempts to characterize instances admitting a "meaningful" clustering. The three main definitions used by theoreticians are Distribution Stability [34], Spectral Separability [222], and Perturbation Resilience [64].

We show that Local Search performs well on the instances with the aforementioned stability properties. Specifically, for the k-means and k-median objective, we show that Local Search exactly recovers the optimal clustering if the dataset is $3 + \varepsilon$-perturbation resilient, and computes a $(1+\varepsilon)$ approximation for distribution stability and spectral separability. This implies the first polynomial time approximation scheme for instances satisfying the spectral separability condition. Our results in turn also support the legitimacy of the stability conditions: They characterize some of the structure of real-world instances that make Local Search a popular heuristic.

1.4 Publications

This thesis is based on the following publications. All authors contributed equally.

- Chapter 3 is based on [79]

  Marc Bury and Chris Schwiegelshohn. Sublinear Estimation of Weighted Matchings in Dynamic Data Streams. European Symposium of Algorithms (ESA), 2015.

  and on

  Marc Bury, Elena Grigorescu, Andrew McGregor, Morteza Monemizadeh, Chris Schwiegelshohn, Sofya Vorotnikova, and Samson Zhou. Structural Results on Matching Estimation with Applications to Streaming. Manuscript.

- Chapter 4 is based on

  Marc Bury, Chris Schwiegelshohn, and Mara Sorella. Sketch 'em all: Approximate Similarity Search on Dynamic Data Streams. Manuscript.

- Chapter 5 is based on [114]

  Vincent Cohen-Addad, Chris Schwiegelshohn, and Christian Sohler. Diameter and k-Center in Sliding Windows. International Colloquium on Automata, Languages, and Programming, (ICALP) 2016.

- Chapter 6 is based on [80]

  Marc Bury and Chris Schwiegelshohn. On Finding the Jaccard Center. International Colloquium on Automata, Languages, and Programming, (ICALP) 2017.

- Chapter 7 is based on [113]

  Vincent Cohen-Addad and Chris Schwiegelshohn. On the Local Structure of Stable Clustering Instances. Symposium on Foundations of Computer Science, (FOCS) 2017.

2 Preliminaries

In this chapter we will introduce the notation, tools, and known results that we will use throughout this thesis.

We refer to sets as a collection of specified objects. The set of real numbers will be denoted by $\mathbb{R}$, the set of non-negative real numbers by $\mathbb{R}_{\geq 0}$ and the set of d-dimensional real vectors by $\mathbb{R}^d$. For two real numbers $a \leq b$, we shorthand $\{x | a \leq x \leq b\} \cap \mathbb{R}$ by $[a, b]$. The set of the first n natural numbers is denoted by $[n]$, i.e. $[n] = \{1, \ldots, n\}$. Given an n by d matrix A, we let A_i denote the ith row, A^j the jth column and $A_{i,j}$ the element in the intersection of the ith row and jth column where $i \in [n]$ and $j \in [d]$.

For a finite set U, we denote by $\mathcal{P}(U)$ the set of all subsets of U. We further denote the symmetric difference of two sets $A, B \subseteq U$ by $A \triangle B := (A \setminus B) \cup (B \setminus A)$ and the negated set by $\overline{A} := U \setminus A$. For some arbitrary but fixed ordering of the elements of U, we define the characteristic $|U|$-dimensional vector a of an item set A by $a_i = 1$ if $U_i \in A$ and $a_i = 0$ otherwise. We will occasionally use $\tilde{O}(f(n))$ to hide factors polylogarithmic in $f(n)$, e.g. $n^2 \log^3 n \in \tilde{O}(n^2)$. For boolean vectors $x, y \in \{0,1\}^n$, denote by $x \oplus y$ a boolean vector where the entry $(x \oplus y)_i = 0$ if and only if $x_i = y_i$.

2.1 Probability

Definition 2.1.1 (Finite Probability Spaces)**.** A finite set Ω is called a *sample space*. The elements of Ω are *elementary events* and a subset of Ω is an event. The probability measure $\mathbb{P}$ is a function $\mathbb{P} : \mathcal{P}(\Omega) \to [0, 1]$ such that $\mathbb{P}[E] = \sum_{x \in E} \mathbb{P}[x]$ for any event $E \subseteq P$ and the sum of all probabilities of elementary events is normalized to be 1, i.e. $\mathbb{P}[\Omega] = \sum_{e \in \Omega} \mathbb{P}[e] = 1$.

We will be particular interested in binary random variables X_E, where X_E is set to 1 if the event E occurs and is set to 0 otherwise. Identically, independently and uniformly distributed random bits are known to be difficult to store. If we view randomness as a resource, for instance in the context of space efficient computation, we require more limited randomness known as k-wise independence. Formally, n binary random variables $X_1, \ldots X_n$ are called *k-wise independent* if for any set of k distinct indices $i_1, \ldots, i_k \in [n]$ and any set of possible outcomes $o_1, \ldots o_k \in \{0, 1\}$ we have

$$\mathbb{P}[X_{i_1} = o_1, \ldots, X_{i_k} = o_k] = 2^{-k}.$$

The expectation of a non-negative discrete random variable X is $\mathbb{E}[X] = \sum_{i \geq 0} \mathbb{P}[X = i] \cdot i$. The variance of X is $\mathbf{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$. The expectation is a linear function, i.e. for any two random variables X_1 and X_2 and any two constants c_1 and c_2 we have $\mathbb{E}[c_1 \cdot X_1 + c_2 \cdot X_2] = c_1 \cdot \mathbb{E}[X_1] + c_2 \cdot \mathbb{E}[X_2]$. If X_1 and X_2 are independent, we further have $\mathbb{E}[X_1 \cdot X_2] = \mathbb{E}[X_1] \cdot \mathbb{E}[X_2]$. For the variance, we have $\mathbf{Var}[c_1 \cdot X_1 + c_2 \cdot X_2] = c_1^2 \cdot \mathbf{Var}[X_1] + c_2^2 \cdot \mathbf{Var}[X_2] + 2c_1c_2 \cdot \mathbf{Cov}(X_1, X_2)$, where the covariance is defined as $\mathbf{Cov}(X_1, X_2) = \mathbb{E}[(X_1 - \mathbb{E}[X_1])(X_2 - \mathbb{E}[X_2])] = \mathbb{E}[X_1 \cdot X_2] - \mathbb{E}[X_1] \cdot \mathbb{E}[X_2]$. We note that if X_1 and X_2 are at least 2-wise independent, $\mathbf{Cov}(X_1, X_2) = 0$.

We will also use the following result due to Pagh and Pagh [258].

Theorem 2.1.2 (Theorem 1.1 of [258]). *Let $S \subset U = \{0, \ldots, u-1\}$ be a set of $k > 1$ elements. For any constants $c > 0$ and $\varepsilon > 0$, and for $1 < v < u$, there is a RAM algorithm that, using time $\lg n (\lg v)^{O(1)}$ and $O(\lg n + \lg\lg u)$ bits of space, selects a family $\mathcal{H}$ of functions from U to $V = \{0, \ldots, v-1\}$ (independent of S) such that:*

- *$\mathcal{H}$ is k-wise independent when restricted to S, with probability $1 - O(\frac{1}{n^c})$.*
- *A function in $\mathcal{H}$ can be represented by a RAM data structure using space $(1+\varepsilon)k \lg v + O(k)$ bits such that function values can be computed in constant time. The data structure of a random function in H can be constructed in time $O(n)$.*

We will use the following concentration bounds taken from [252].

Theorem 2.1.3 (Markov's Inequality). *Let X be a non-negative random variable and let $c > 0$ be a constant. Then*

$$\mathbb{P}[X > c] \leq \frac{\mathbb{E}[X]}{c}.$$

Theorem 2.1.4 (Chebyshev's Inequality). *Let X be a non-negative random variable and let $c > 0$ be a constant. Then*

$$\mathbb{P}[|X - \mathbb{E}[X]| > c] \leq \frac{\boldsymbol{Var}[X]}{c^2}.$$

Theorem 2.1.5 (Multiplicative Chernoff Bound). *Let $X_1, \ldots X_m$ be binary random variables with $\mu = \mathbb{E}[\sum_{i=1}^m X_i]$. Then for any $0 < \delta < 1$*

$$\mathbb{P}\left[\sum_{i=1}^m X_i > (1+\delta)\cdot\mu\right] \leq \exp\left(-\frac{\delta^2 \cdot \mu}{3}\right) \quad \textit{and} \quad \mathbb{P}\left[\sum_{i=1}^m X_i < (1-\delta)\cdot\mu\right] \leq \exp\left(-\frac{\delta^2 \cdot \mu}{2}\right).$$

2.2 Metric Spaces

We start with the definition of metric spaces.

Definition 2.2.1. Let X be a set and let $\text{dist} : X \times X \to \mathbb{R}_{\geq 0}$ be a function. Then (X, dist) is a *metric space* if the following three conditions hold for any $x, y, z \in X$:

- $\text{dist}(x, y) = 0 \Leftrightarrow x = y$
- $\text{dist}(x, y) = \text{dist}(y, x)$
- $\text{dist}(x, y) \leq \text{dist}(x, z) + \text{dist}(z, y)$.

We also say that dist is a *distance function*. The inequality is also known as the *triangle inequality*. We define the aspect ratio $\alpha = \frac{\max_{p \neq q \in X} \text{dist}(p,q)}{\min_{p \neq q \in X} \text{dist}(p,q)}$ if it is defined. If the aspect ratio is not defined the metric space consists of a single point.

We will consider two special cases of metric spaces. The first is induced by ℓ_p norms such as Euclidean spaces. The others, defined in the following, are those induced by rational set similarities.

Definition 2.2.2 (Similarity Functions)**.** Let U be a finite set.

- A symmetric function $S : \mathcal{P}(U) \times \mathcal{P}(U) \to [0, 1]$ with $S(A, A) = 1$ for all $A \in \mathcal{P}(U)$ is a *similarity*.
- Given rational numbers $x, y \geq 0$ and $z' \geq z \geq 0$, the *rational set similarity* $S_{x,y,z,z'}$ between two non-empty sets $A, B \in U$

$$S_{x,y,z,z'}(A, B) = \frac{x \cdot |A \cap B| + y \cdot |\overline{A \cup B}| + z \cdot |A \triangle B|}{x \cdot |A \cap B| + y \cdot |\overline{A \cup B}| + z' \cdot |A \triangle B|}$$

  if it is defined and 1 otherwise.
- Given a rational set similarity $S_{x,y,z,z'}$, the induced distance function is defined as

$$D_{x,y,z,z'} := 1 - S_{x,y,z,z'}(A, B).$$

- Given rational numbers $x, y \geq 0$ and $z' \geq z \geq 0$, the *root similarity* with root $0 < \alpha \leq 1$ is

$$S_{x,y,z,z'}(A, B)^\alpha = 1 - (1 - S_{x,y,z,z'}(A, B))^\alpha.$$

For an overview of rational set similarities, we refer to Gower and Legendre [165]. The arguably most well known rational set similarity is the *Jaccard index* defined as $S_{1,0,0,1}(A, B)$. Since the Jaccard index will be encountered with the greatest frequency throughout this thesis, we will refer to it by S and the *Jaccard distance* by $D(A, B) = 1 - S(A, B)$. We denote numerator and denominator of a rational set similarity by $Num_{x,y,z,z'}(A, B)$ and $Den_{x,y,z,z'}(A, B)$, respectively.

The following characterization of metric distance functions is due to Janssens [201].

Theorem 2.2.3. *Let U be a ground set and $S_{x,y,z,z'}$ be a rational set similarity over subsets of U. Then $1 - S_{x,y,z,z'}$ is a metric if and only if $z' \geq \max(x, y, z)$.*

2.2.1 Nearest Neighbor Search

One of the most important computational challenges in metric spaces is to quickly find points with small distances or, conversely, of high similarity. Despite being a simple problem insofar that it admits a trivial polynomial time solution, nearest neighbor search is often the most time consuming part of an algorithm in practice. The precise task can vary depending on the application. Here, we will focus on the Jaccard similarity, while touching on some of the more general aspects of locality sensitive hashing. For an overview of locality sensitive hashing, we refer to Andoni and Indyk [17].

We are given n item sets and aim to provide a data structure such that all pairs of item sets with similarity at least T are found, where T is a parameterized threshold. The most straightforward approach is to evaluate the similarity of all pairs, which requires $\binom{n}{2} \in O(n^2)$ distance evaluations.

We will be aiming for a linear or near linear number of distance evaluations, while relaxing the problem. Instead of using an exact threshold T, we might be contend with finding high similarity pairs while rejecting low similarity pairs. One way to formalize such requirements is via locality sensitive hashing originally defined by Indyk and Motwani [196] (see also the follow up paper by Gionis, Indyk and Motwani [162]) as follows.

Definition 2.2.4. Given a metric space (X, dist), a set B of buckets and a family $\mathcal{F}$ of hash functions $h : X \to B$ and a distribution over F, we say that F is (r_1, r_2, p_1, p_2)-sensitive if for any two points $a, b \in X$

- $\mathbb{P}_{h\sim\mathcal{F}}[h(a) = h(b)] \geq p_1$ if $\text{dist}(a, b) \leq r_1$ and
- $\mathbb{P}_{h\sim\mathcal{F}}[h(a) = h(b)] \leq p_2$ if $\text{dist}(a, b) \geq r_2$.

For similarity functions, Charikar [90] considered the following special case.

Definition 2.2.5. Let $S : U \times U \to [0, 1]$ be a similarity function. We say that S is *Locality Sensitive Hashable* (or LSHable) if there exists a set of buckets B and a family $\mathcal{F}$ of hash functions $h : U \to B$ with some associated distribution over $\mathcal{F}$ such that for any two elements $a, b \in U$ we have

$$\mathbb{P}_{h\sim\mathcal{F}}[h(a) = h(b)] = S(a, b).$$

The definition by Indyk and Motwani is more general and is more convenient to present the results of this thesis. Nevertheless, Charikar's definition has a number of appealing properties that we will discuss in the following. It should be noted that the notions behind the two definitions are the same.

By examining the properties of distance functions induced by a similarity S, Charikar [90] discovered a surprisingly simple relationship to LSHability.

Lemma 2.2.6. *Let $S : U \times U \to [0, 1]$ be a similarity function. If S is LSHable, then $1 - S$ is a metric.*

This lemma gives some indication as to whether or not S is LSHable or not. For rational set similarities, Chierichetti and Kumar [96] showed this condition in fact to be sufficient which, together with Theorem 2.2.3 by Janssens [201], gives us:

Theorem 2.2.7. *Let $S_{x,y,z,z'}$ be a rational set similarity. Then the following three conditions are equivalent.*

- *$S_{x,y,z,z'}$ is LSHable.*
- *$D_{x,y,z,z'}$ is a metric.*
- *$z' \geq \max(x, y, z)$.*

From a historical perspective, the Jaccard similarity $|A \cap B|/|A \cup B|$ was the first similarity (retroactively) known to be LSHable using min-wise independent hash functions, see the pioneering work by Broder et al. [73, 74, 76]. Roughly speaking, min-hashing computes a fingerprint of a binary vector by permuting the entries and storing the first non-zero entry. In practice, a random hash function satisfying certain conditions is sufficient instead of a random permutation of the entries. When looking for item sets similar to some set A, one can arrange multiple fingerprints to filter out sets of small similarity while retaining sets of high similarity, see Chapter 4 for more details and also the original reference by Cohen et al [106].

Many papers focused on the design and analysis of random hash functions, see for instance [75, 193, 144]. While min-wise independent hash functions give the best performance in theory, they are often considered infeasible to store. Thorup [274] showed that the more space efficient 2-wise independent hash functions work well. Other work focused on the efficiency of computing fingerprints. For instance, a faster estimation of similarity is possible by storing the k smallest non-zero entries, see Cohen and Kaplan [107, 108]. Li and König [229] introduced b-bit hashing, where each fingerprint is represented by b bits, which was shown to be space-optimal by Pagh et al [259]. Min-hashing is also featured in other computational models such as parallel algorithms [273], sliding windows [125], and distributed frameworks like MapReduce [285].

2.3 Linear Algebra

The other metric space that will feature prominently in this thesis is high dimensional Euclidean space. We start by giving the definition of a vector norm.

Definition 2.3.1 (Vector Norms)**.** Given a vector space X over the real number $\mathbb{R}$, a norm f is a function satisfying the following three properties:

- $f(ax) = |a| \cdot f(x)$ for all $x \in X$ and $a \in \mathbb{R}$
- $f(x + y) \leq f(x) + f(y)$ for all $x, y \in X$
- $f(x) = 0 \Rightarrow x = 0$.

Among the most common examples of a vector norm are ℓ_p norms.

Definition 2.3.2 (ℓ_p Norms)**.** Let $x \in \mathbb{R}^d$. Then for any $p > 0$, the ℓ_p norm of x is defined as $||x||_p = \sqrt[p]{\sum_{i=1}^{d} |x_i|^p}$. For $p = 0$ we define $||x||_0 = |\{x_i \neq 0, i \in [d]\}|$.

The well known Euclidean norm is the special case $p = 2$. The Euclidean distance between two points $x, y \in \mathbb{R}^d$ is $||x-y||_2$. Two vectors x, y are called *orthogonal* if $x^T y := \sum_{i=1}^{d} x_i \cdot y_i = 0$. A matrix is diagonal if $n = d$ and $A_{i,j} = 0$ for all $i \neq j$ and $i, j \in [n]$. The identity matrix I of dimension n is the n by n diagonal matrix whose diagonal entries are all 1. A matrix A has *orthogonal rows* (resp. columns) if $AA^T = I$ (resp. $A^T A = I$).

Definition 2.3.3 (Singular Value Decomposition)**.** Let $A \in \mathbb{R}^{n \times d}$ be an n by d matrix with real values. Further assume that $d \leq n$. The *singular value decomposition* (SVD) are three matrices $U \in \mathbb{R}^{n \times d}$, $\Sigma \in \mathbb{R}^{d \times d}$ and $V \in \mathbb{R}^{d \times d}$ with $A = U\Sigma V^T$ such that

- U has orthogonal columns, i.e. $U^T U = I$,
- Σ is a diagonal matrix with non-negative entries, and
- V has orthogonal rows, i.e. $VV^T = I$.

The diagonal entries of Σ are called *singular values*, the columns of U *left singular vectors* and the rows of V *right singular vectors*. A fundamental theorem of linear algebra is that every matrix has a singular value decomposition which is unique up to perturbation of columns and rows. By convention, the elements of Σ are usually ordered in descending order, i.e. $\Sigma_{1,1} := \sigma_1 \geq \Sigma_{2,2} := \sigma_2, \ldots \geq \ldots \Sigma_{d,d} := \sigma_d \geq 0$. There exist several connections between the singular value decomposition and matching, as well as the Euclidean k-means problem which we will remark on in the following sections.

The SVD allows us to generalize the notion of ℓ_p norms to matrices.

Definition 2.3.4 (Schatten Norms)**.** Let $A \in \mathbb{R}^{n,d}$ be an n by d matrix with real values with singular value decomposition $A = U\Sigma V^T$ and let $p > 0$. Further assume that $d \leq n$. Then the pth Schatten norm is defined as $||A||_{S0} = \left(\sum_{i=1}^{d} \sigma_i^p\right)^{1/p}$. For $p = 0$ we define $||A||_{S0} = |\{\sigma_i \neq 0\}|$.

Some well known special cases are $p = 0$ which is known as the *rank* of A, $p = 2$ which is equivalent to the Frobenius norm $||A||_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{d} A_{i,j}^2\right)^{1/2}$ and $p = \infty$ which is known as the spectral norm $||A||_2 = \max_{x\neq 0}\frac{||Ax||_2}{||x||_2} = \sigma_1$. We note that both the rank of a matrix and the number of non-zero elements of a vector are strictly speaking not norms as they do not satisfy scalability, but they are often referred to as such.

One of the most extensively studied topics in numerical linear algebra is the computation of low-rank matrices that are in some sense similar to the original matrix. Specifically, given an n by d real matrix A, the task is to find an n by d real matrix $\hat{A}$ of rank at most k, such that $||A - \hat{A}||_F$ is minimized. Alternatively, one may consider the equivalent formulation of finding an n by k matrix X with orthogonal columns such that $||A - XX^TA||_F$ is minimized.

The optimal solution of this problem is implicit in the SVD: $A_k = U\Sigma_k V^T$ where Σ_k is the truncated diagonal matrix containing the largest k singular values. For the alternative problem formulation, the optimal projection matrix X consists of the first k columns of U. A low rank approximation with respect to other norms than the Frobenius norm have also been considered, for recent hardness results see Clarkson and Woodruff [105], though the Frobenius norm is by far the best studied, understood and arguably useful variant. It should be noted that $\hat{A}$ is also the optimum solution for any norm invariant under rotations which is, for instance, the case with the spectral norm.

Computing an SVD can be done (up to numerical precision) in time $O(n \cdot d \cdot \min(n, d))$, see Trefethen and Bau [276] for an overview. If we are only interested in a low rank approximation, faster approximate algorithms exist, see for example Clarkson and Woodruff [103, 104], Sarlós [267], and Deshpande and Vempala [128]. A complete review of the computational aspects of the SVD is out of the scope of this thesis; the main take away message is that the computation is considered feasible.

2.4 Graphs

Definition 2.4.1 (Graphs)**.**

- An undirected graph $G(V, E)$ consists of a set of n *nodes* V and a set E of 2-element subset of V called *edges*.
- A weighted graph $G(V, E, w)$ is a graph with a weight function $w : E \to \mathbb{R}$.
- A graph $G'(V', E')$ is a *subgraph* of $G(V, E)$ if $V' \subseteq V$ and $E' \subseteq (V' \times V') \cap E$.
- For a subset of nodes $U \subseteq V$, we call $(U, (U \times U) \cap E)$ the *induced subgraph* of U.

We will only consider undirected graphs, i.e. the edge (x, y) is identical to the edge (y, x). The same also holds for weighted graphs, i.e. $w(x, y) = w(y, x)$. Two nodes $u, v \in V$ are *adjacent*, if $(u, v) \in E$. The edge (u, v) is *incident* to the nodes u and v.

The two classic graph problems studied in this thesis are matching and the more loosely defined clustering tasks.

2.4.1 Matching

Definition 2.4.2 (Matching)**.** Let $G(V, E)$ be a graph.

- A *matching* is a set of edges $M \subseteq E$ such that $e_1 \cap e_2 = \emptyset$ for any two edges $e_1, e_2 \in M$.
- A *maximal matching* is a subset of edges $M \subseteq E$ such that there exists no edge $e \in E \backslash M$ with $M \cup \{e\}$ being a matching.
- A *maximum matching* is a maximal matching of maximum cardinality.

The currently fastest known combinatorial algorithm with running time $O(\sqrt{|V|} \cdot |E|)$ for unweighted matching is due to Micali and Vazirani [250], see also an earlier algorithm with the same time bound by Hopcroft and Karp [190] for bipartite graphs. For sufficiently sparse graphs, Madry [237] recently gave an improved algorithm running in time $\tilde{O}(|E|^{10/7})$.

There also exists a rich body of work on algebraic aspects of matching, which is particularly relevant for this thesis. These algorithms are based on the Tutte-matrix of a graph $G(V, E)$ defined as

$$T_{i,j} = \begin{cases} x_{i,j} & \text{if } i > j \text{ and } (i,j) \in E \\ -x_{i,j} & \text{if } j > i \text{ and } (i,j) \in E \\ 0 & \text{if } (i,j) \notin E, \end{cases}$$

where $x_{i,j}$ are indeterminates. In his seminal paper, Tutte [277] showed that a graph contains a *perfect matching*, i.e. a matching of size $n/2$ if and only if for some choice of indeterminates the determinant of T is nonzero. This was later generalized by Lovász [234] as follows.

Theorem 2.4.3. *Let $G = (V, E)$ be a graph with a maximum matching M and Tutte matrix T_G. For an assignment $w \in \mathbb{R}^{|E|}$ to the indeterminates of T_G we denote the matrix by $T_G(w)$ where the indeterminates are replaced by the corresponding assignment in w. Then we have*

$$\max_w \{rank(T_G(w))\} = 2 \cdot |M|.$$

In order to calculate the maximum of the rank, Lovász [234] also showed that the rank of the matrix where the indeterminates are replaced by random numbers uniformly drawn from $\{1, \ldots, R\}$ is equal to $\max_w \{rank(T_G(w))\}$ with probability at least $1 - |E|/R$.

Theorem 2.4.4. *Let $G = (V, E)$ be a graph and $r \in \mathbb{R}^{|E|}$ be a random vector where each coordinate is uniformly chosen from $\{1, \ldots, R\}$ with $R \geq |E|$. Then we have*

$$rank(T_G(r)) = \max_w \{rank(T_G(w))\}$$

with probability at least $1 - |E|/R$.

The simplicity of Theorem 2.4.4 makes it an appealing foundation for most algebraic matching algorithms, see, for instance, [253, 264, 266], though deterministic approaches aim to manipulate the Tutte-matrix directly, see Geelen [160]. The best known algebraic algorithm is due to Harvey [186], who gave an $O(|V|^\omega)$ time algorithm, where ω is the exponent of matrix multiplication. Finding better bounds on ω is still a very active area of research. The currently best bound of roughly 2.3728639 is due to Le Gall [156]. For dense graphs, Harvey's algorithm achieves a better running time than the combinatorial approaches mentioned above.

The notion of matchings can be naturally extended towards weighted graphs

Definition 2.4.5 (Weighted Matching)**.** Let $G(V, E, w)$ be a weighted graph.

- The *weight* of a matching $M \subseteq E$ is defined as $w(M) := \sum_{e \in M} w(e)$.
- A *weighted maximum matching* is a maximal matching $M \subseteq E$ with maximum weight.

Maximum weighted matching admits a polynomial time algorithm, see for instance an algorithm by Lovász and Plummer [235] based on the seminal unweighted algorithm by Edmonds [133]. Recently, for integer weights bounded in $[1, W]$, Duan et al. [132] gave an algorithm running in time $O(m\sqrt{n} \log nW)$, which nearly matches the running time of the Micali and Vazirani algorithm [250] for unweighted matching.

2.5 Clustering

Before defining the clustering problems we consider in this thesis, we want to remark that there exist countless other variants to define and analyze various clustering problems. What they all share the task to partition the set of nodes into (usually disjoint) subsets called clusters. In theory literature, the goodness of a clustering is measured via some objective function. In this thesis, we mainly consider center based objective functions. The most important of these are k-median, k-means, and k-center.

Definition 2.5.1 (k-Clustering)**.** Let A be a set of clients, F a set of centers, *dist* a function $(A \cup F) \times (A \cup F) \to \mathbb{R}_{\geq 0}$ such that $(A \cup F, \text{dist})$ is a metric space and k a non-negative integer. Then for a subset $S \subseteq F$ of k centers, we have the following objective functions.

The *k-median clustering problem* minimizes $\text{cost}(S) = \sum_{x \in A} \min_{c \in S} \text{dist}(x, c)$.

The *k-means clustering problem* minimizes $\text{cost}(S) = \sum_{x \in A} \min_{c \in S} \text{dist}^2(x, c)$.

The *k-center clustering problem* minimizes $\text{cost}(S) = \max_{x \in A} \min_{c \in S} \text{dist}(x, c)$.

The *(k, p)-clustering problem* minimizes $\text{cost}(S) = (\sum_{x \in A} \min_{c \in S} \text{dist}^p(x, c))^{1/p}$ for any $p > 0$.

We say that the *clustering* of A induced by S is the set of subsets $C = \{C_1, \dots C_k\}$ such that $C_i = \{x \in A \mid c_i = \operatorname{argmin}_{c \in S} \text{dist}(x, c)\}$.

We remark that $(k,1)$ clustering is identical to k-median, $(k,2)$ clustering is identical to k-means up to taking a square root of the final objective value and k-center is $\lim_{p\to\infty}(k,p)$ clustering. In the above definitions, A will be the nodes of a graph given as input. We cannot specify F in general. F may be identical to A, i.e. we choose our set of centers from the set of input nodes. Sometimes the set of centers will be far larger than the input, for instance when the underlying metric space has a small description as is the case for the Jaccard metric or Euclidean space. In every section we will specify what the algorithm is given as input and from which set F we draw candidate centers.

If the nodes are points in Euclidean space, we abandon the graph-based view of the problem in favor of an algebraic one. In this case, we view the input as an n by d matrix A, where A_i is the ith point of the dataset. The k-means objective function is especially interesting in this case due to the relationship with low rank approximations. The connection is made via the following well known fact.

Fact 2.5.2 (Cohen et al. [109])**.** *Let A be a set of points in Euclidean space and denote by $c(A) = \frac{1}{|A|}\sum_{x\in A} x$ the* centroid *of A. Then the 1-means cost of any candidate center c can be decomposed via*

$$\begin{aligned}\sum_{x\in A}||x-c||^2 &= \sum_{x\in A}||x-c(A)||^2 + |A|\cdot||c(A)-c||^2 \\ &= \frac{1}{2\cdot|A|}\sum_{x\in A}\sum_{y\in A}||x-y||^2 + |A|\cdot||c(A)-c||^2.\end{aligned}$$

Note that the centroid is the optimal 1-means center of A. If the set of centers are centroids of their respective clusters, we can rewrite the objective function in matrix form by using the cluster matrix $X \in \mathbb{R}^{n\times k}$ with

$$X_{i,j} = \begin{cases}\frac{1}{\sqrt{|C_j|}} & \text{if } A_i \in C_j \\ 0 & \text{else}\end{cases}$$

to denote membership. It is easy to see that the sum of all the entries of a vector can be determined by projecting onto the all 1-vector, and that the centroid can be obtained by appropriately rescaling the projection. It follows that $(XX^TA)_i = c(C_j)$ if and only if $A_i \in C_j$. X is an orthogonal projection, i.e. $X^TX = I$ and $||A - XX^TA||_F^2$ is the cost of the induced k-means clustering. Euclidean k-means is therefore a constrained rank k-approximation problem. The fact that $||A - A_k||_F^2 \leq ||A - XX^TA||_F^2$ for any clustering matrix X is frequently used for dimension reduction, see for instance [66, 109, 131, 147, 222] and Chapter 7.

Clustering is a very active area of research and it is impossible to cover all relevant aspects even if we restrict ourselves to center-based clustering. In the following we will only mention the state of the art hardness and approximability results.

In arbitrary finite metric spaces ($F = A$), there exist lower bounds of $1+2/e$ (resp. $1+3/e$) on the approximation ratio for the k-median (resp. k-means problem) unless $P = NP$, see Guha and Khuller [171] and Jain et al. [198]. For the k-center problem, Gonzalez [164] gave a 2-approximation algorithm and also showed that this bound is tight, unless $P = NP$, see also Hochbaum and Shmoys [189]. The current best polynomial time approximation factor for metric k-median is $2.611 + \varepsilon$, see the algorithm of Byrka et al. [81] which is based on the earlier $(1+\sqrt{3}+\varepsilon)$-approximation algorithm of Li and Svensson [230] and 4-approximation of Jain et al. [198]. Constant approximation factors for metric k-means appear in various papers due to the fact that squared distances admit an approximate triangle inequality which can often be folded into the approximation factor, see for instance [199, 249]. The best known approximation ratio 6.357 is due to Ahmadian et al. [7].

In Euclidean spaces, these problems are non-trivial even using only one center. 1-means can be solved analytically via Fact 2.5.2. Closed formulas do not exist for the 1-median or 1-center, but these problems can be solved to arbitrary numerical precision in polynomial time via convex programming [169]. Recently, Cohen et al. [110] gave a nearly linear time algorithm for the 1-median problem in arbitrary dimensions. Linear or nearly linear time algorithms for the 1-center problem currently do not have a polynomial dependency on the dimension, see Matousek et al. [241] and the references therein.

If k and d are part of the input, all three problems are APX-hard, see Guruswami and Indyk [178] for k-median, Awasthi et al. [36] for k-means, and Feder and Greene [140] for k-center. By constraining the candidate centers to be input points, the approximation algorithms for k-median and k-means can be used for the Euclidean case while losing an additional factor of 2, though some papers bypass this by computing a small set of candidate centers, see, for instance, Kanungo et al. [208] and Matousek [240]. The algorithm by Gonzalez [164] retains its 2-approximation factor guarantee in Euclidean spaces. Despite the inapproximability results, various flavors of $(1+\varepsilon)$-approximation algorithms are known in Euclidean space for either fixed k or fixed d. When d is fixed, Arora et al. gave the first PTAS [25] for the k-median problem. This result was subsequently improved to an efficient PTAS by Kolliopoulos and Rao [218] and Har-Peled et al. [184, 185]. For k-means, two recent results have independently discovered a PTAS for various restricted metrics including constant dimensional Euclidean space, see Cohen-Addad et al. [111] and Friggstad et al [154]. For k-center, the aforementioned APX hardness result for approximation factors less than $(1+\sqrt{7})/2 \approx 1.82$ of Feder and Greene [140] holds even if $d = 2$. When k is fixed but d is arbitrary, the best current algorithm for both k-median and k-means is due to Feldman and Langberg [145] with a running time of $O(nd + \exp(\text{poly}(k, \varepsilon^{-1})))$. For k-center, the only known $(1+\varepsilon)$-approximation running in time $O(nd + \exp(\text{poly}(k, \varepsilon^{-1})))$ is due to Bădoiu et al. [45]. We note that various other approximation algorithms exist that constrain the input in more subtle ways. One such possibility are stability assumptions which we describe in more detail in Chapter 7.

2.6 Streaming Model

Definition 2.6.1 (Data Streams)**.** Let $X[1], \dots X[m]$ be an array corresponding to the entries of a vector, wich is initial $\mathbf{0}$. A stream consists of a sequence of additive updates (i, j) were $i \in [m]$ denotes the entry of the array and j the number added to $X[i]$.

- An *insertion-only stream* is a sequence S of pairs (i, j) such that every j is non-negative and for every $i \in [m]$ there exists only one pair (i, j).
- A *sliding window stream* with window size N is an insertion only stream where at any given time we only consider the most recent N elements of S. The *Time To Live* (TTL) of an element p is initially N and gets decremented with each subsequently inserted element. An item p *expires* if $TTL(p) = 0$.
- A *strict turnstile stream* is a sequence S of pairs such that for all pairs P with entry $i \in [m]$, we have $\sum_{(i,j)\in P} j \geq 0$.
- A *turnstile stream* is an arbitrary sequence S of pairs.

We will sometimes abuse notation and extend TTL to negative numbers to indicated the number of elements submitted after expiration i. e., $TTL(p) = -10$ means that 10 elements were submitted after the expiration of p.

A streaming algorithm aims to compute some function $f : S \to \mathbb{R}$ using as little space as possible. Few problems allow for an exact computation, giving rise to approximations. The output $Alg(S)$ of a streaming algorithm Alg is required to satisfy $f(S) \leq Alg(S) \leq \alpha \cdot f(S)$ for some approximation factor $\alpha > 1$. If Alg is randomized, we require the output to satisfy the above guarantee with probability greater than $1/2$.

In addition to the approximation factor, we measure the quality of a streaming algorithm via the space requirement. Depending on the problem, the space requirement is given in terms of bits, words, points, or size of an induced subgraph. Space bounds in terms of bits are usually the strongest and most general, while the other space bounds are used for constrained algorithms or conditional lower bounds. If possible, a streaming algorithm will aim for a space bound polylogarithmic in the input size. A secondary objective is to minimize the processing time either amortized over all updates or as the worst-case update time for each update individually.

Classic problems studied in data-streams include the approximation of vector statistics such as ℓ_p norms, entropy and frequent item mining. The paper that set the foundations of most modern theoretical streaming research is the seminal 1996 paper by Alon, Matias, and Szegedy [15], though the models antecedents date back even further. Sliding windows were introduced later in 2002 by Datar, Gionis, Indyk and Motwani [124]. Since then, streaming research has branched out into learning, optimization, computational geometry and the comparatively recent model of graph streaming. The field is still evolving rapidly and to the best

of our knowledge there exists no state of the art survey. The interested reader may want to see Muthukrishnan [254] for a good, if slightly outdated overview.

2.6.1 Graph Streams

Definition 2.6.2 (Unweighted Graph Streams)**.** Let $G(V, E)$ be an initially empty undirected graph.

- An *insertion-only graph stream* is a sequence S of pairs $(u, v) \in V \times V$ signifying that (u, v) has been added to E.
- A *dynamic graph stream* is a sequence S of triples $(u, v, t) \in V \times V \times \{0, 1\}$ with $t = 1$ signifying that (u, v) has been added to E and $t = 0$ signifying that (u, v) has been deleted from E.
- A dynamic edge stream is *consistent* if any update (u, v, t) implies that $(u, v) \in E$ if $t = 0$ and $(u, v) \notin E$ if $t = 1$.

The number of nodes is n and the stream length is either $|E|$ for insertion-only streams or unbounded for dynamic graph streams. We will always assume our streams to be consistent. We emphasize that the graph is unweighted, i.e. (u, v) is identical to (v, u). Note that the consistency assumption for dynamic graph streams is equal to strict turnstile streams with $j \in \{-1, 1\}$. We also extend these definition to weighted graphs as follows.

Definition 2.6.3 (Weighted Graph Streams)**.** Let $G(V, E, w)$ be an initially empty undirected weighted graph with $w \in E \to [W]$.

- An *insertion-only weighted graph stream* is a sequence S of pairs $(u, v, w(u, v)) \in V \times V \times [W]$ signifying that (u, v) with weight $w(u, v)$ has been added to E.
- A *weighted dynamic graph stream* is a sequence S of quadruples $(u, v, t, w(u, v)) \in V \times V \times \{0, 1\} \times [W]$ with $t = 1$ signifying that (u, v) with weight $w(u, v)$ has been added to E and $t = 0$ signifying that the edge (u, v) with weight $w(u, v)$ has been deleted from E.
- A dynamic weighted edge stream is *consistent* if any update (u, v, t, w') implies that $(u, v) \in E$ with weight $w' = w(u, v)$ if $t = 0$ and $(u, v) \notin E$ if $t = 1$.

A dynamic weighted and consistent graph stream is weaker than the strict turnstile as an edge deletion is required to supply the edge weight. For instance if the edge (u, v) currently has weight 2, and we want it to have weight 5, we would have to first process the update $(u, v, 0, 2)$ and then process $(u, v, 1, 5)$, as opposed to merely adding 3 units of weight to the edge (u, v). We note that to store the weight of an edge, we require $\log W$ bits.

Aside from matching, which we will discuss later in detail, problems studied in graph streams include spanners [55, 134, 214], connectivity [11, 172, 261], diameter and shortest paths [142, 179], subgraph counting [54, 72, 78, 203, 247], clique and independent set [135, 181], minimum spanning tree [141, 11], vertex cover [100], cut problems [9, 12, 211, 212], spectral sparsification [213, 214, 215], dense subgraph detection [61, 245], and other clustering problems [8]. The field was founded in an influential technical report from 1998 by Henzinger, Raghavan, and Rajagopalan [188] who showed that even simple problems require up to $\Omega(n^2)$ space. It wasn't until 2004 when Feigenbaum, Kannan, McGregor, Suri and Zhang [141] introduced the *semi-streaming model* that the field really took off. Unlike the standard streaming model, few graph problems admit streaming algorithms using at most polylogarithmic or even sublinear (in n) space. The semi-streaming model is more flexible in that it allows $O(n \cdot \text{polylog}(n))$ bits of space and possibly multiple passes. The model has since become quite popular and it is now encountered well beyond graph problems such as optimization [10, 102], hypergraph problems such as set cover [31, 84, 127, 136, 183] and indeed the locality sensitive hashing paper by Bury and the author (Chapter 4). Dynamic graph streams are a comparatively recent model introduced by Ahn, Guha and McGregor [11, 12] in two papers in 2012. We note that the model is distinct from, though related to the field of dynamic graphs. Here we are similarly given a sequence of edge insertions and deletions, but the focus lies more on fast update times than space constraints. To the best of our knowledge, sliding window graph streams have only been considered in one paper by Crouch, McGregor, and Stubbs [117] in which they consider a variety of problems including matching, spanners, and sparsification.

Matching in Datastreams The arguably most studied problem in graph streaming is matching. Maintaining a 2-approximation to the maximum matching in an insertion-only stream can be straightforwardly done by greedily maintaining a maximal matching [141]. Despite being an almost trivial algorithm, no better algorithm is known. In fact, lower bounds due to Goel et al. [163] showed that no algorithm using $\tilde{O}(n)$ space can achieve an approximation ratio better than $\frac{3}{2}$ which was improved by Kapralov to $\frac{e}{e-1}$ [209]. Their technique is of particular interest and all lower bounds for maintaining matchings are derived from it in some measure. An *ε-Ruzsa-Szeméredi* graph is a graph whose edges can be partitioned into sets of matchings of size $\varepsilon \cdot n$. The general idea is to (1) first submit a Ruzsa-Szeméredi graph of appropriate parametrization, (2) pick one of the matchings at random, (3) submit edges such that the remaining nodes are matched. By showing that the algorithm has to store at least a constant number of bits per edge and giving a construction for $1/2 - \delta$-Ruzsa-Szeméredi graphs on $2n$ nodes with $n^{1+\Omega(1/\log\log n)}$ edges, Goel et al. [163] proved their lower bound. Kapralov [209] further generalized this construction by embedding it into multiple phases.

Konrad [219] used yet a different construction of Ruzsa-Szeméredi graphs to obtain lower bounds for an n^{ε} approximate matching. He showed that the any linear sketched based

algorithm requires $\Omega(n^{3/2-4\varepsilon})$ bits of space. Since linear sketches can be used to obtain lower bounds of turnstile streams [232] and dynamic graphs [13], this result gives a lower bound for maintaining approximate matchings in dynamic graphs. Subsequently, Assadi et al. [33] further improved this lower bound by using an improved construction of Ruzsa-Szeméredi. The construction due to Alon et al. [16] consists of $\binom{n}{2} - o(n^2)$ edges which can be decomposed into matchings of size $n^{1-o(1)}$. Both Konrad and Assadi et al. gave nearly tight (up to polylogarithmic factors) upper bounds for dynamic graph streams by randomly sampling edges. Chitnis et al. [100] showed that small matchings of size k can be maintained using $\tilde{O}(k^2)$ space in dynamic streams. For the somewhat related question of approximating a matching in a distributed setting on k sites, Huang et al. [191] gave a lower bound of $\Omega(k \cdot n/\alpha^2)$ for any approximation factor $\alpha > 1$ using techniques from Phillips et al. [261]. Their result also implies a lower bound of $\Omega(n/\alpha^2)$ for dynamic streams.

If the edges of an insertion only stream are assumed to arrive in random order as opposed to an adversarial order, Konrad et al. [220] were able to obtain an approximation factor of 1.989, using $\tilde{O}(n)$ space. In sliding window streams, Crouch et al. [117] gave a $(3+\varepsilon)$-approximation using the smoothed histogram technique of Braverman and Ostrovski [71].

For weighted matching (MWM), a series of results have been published for insertion streams [118, 141, 243, 137, 138, 288] with the current best bound of $(2 + \varepsilon)$ being due to Paz and Schwartzman [260]. Many of these algorithms operate in an even more restricted preemptive online model. Here, the edges of a graph arrive one by one and each edge can either be added to the matching or discarded. If the edge is added, it may be deleted at a later date. Once an edge gets deleted, it cannot be recalled by the algorithm. The model gives rise to lazy update procedures often found in online algorithms. For deterministic online preemptive matching algorithms a 5.828 approximation was proven to be optimal [280]. Randomized algorithms are difficult to use, as independent coin tosses for each step yield no improvement over deterministic algorithms [99]. Epstein et al. [138] showed that by randomly rounding edge weights and using the deterministic update routine, one can obtain an 5.356 approximation. To obtain better bounds, algorithms maintaining multiple matchings have been proposed. Here, the weights are partitioned into exponentially growing ranks and maintaining a maximum matching for each rank. Grigorescu et al. [168] proved that this algorithm achieves a $(3.5 + \varepsilon)$-approximation factor. A very elegant algorithm by Paz and Schwartzman [260] with an approximation factor of $(2 + \varepsilon)$ was recently published. Essentially, any further improvement for weighted matching also implies an improved for the trivial greedy algorithm for unweighted matching.

A number of papers also dealt with multi-pass algorithms for the matching problem. Guruswami and Onak [179] showed that determining the maximum matching size in p rounds requires $n^{1+\Omega(1/p)}/p^{O(1)}$ bits of space. The aforementioned paper by Konrad et al. [220] also showed that 2 passes are sufficient to beat a 2-approximation ratio in adversarially or-

dered insertion-only streams. McGregor [243] showed that a $(1+\varepsilon)$-approximation can be computed in $\varepsilon^{-1/\varepsilon}$ passes while using $\tilde{O}(n)$ space. For weighted matching, he obtained a $(2+\varepsilon)$-approximation in ε^{-3} passes. Ahn and Guha [10] subsequently improved this to a $(3/2+\varepsilon)$-approximation using $O(n \log n/\varepsilon)$ space and $\varepsilon^{-2}\log 1/\varepsilon$ passes and a $(1+\varepsilon)$-approximation using $O(n \log n/\varepsilon^{-4})$ space and $\varepsilon^{-4}\log n$ passes. Building on work by Lattanzi et al. [227] and Jowhari et al. [204], Ahn et al. [12] also obtained a $O(n^{1+1/p}\cdot \text{poly}(\varepsilon^{-1}))$ space dynamic graph streaming algorithm using $O(p\cdot\varepsilon^{-2}\log\log\varepsilon^{-1})$ passes in the unweighted case and $O(p\cdot\varepsilon^{-2}\log\varepsilon^{-1})$ passes in the weighted case.

To bypass the natural $\Omega(n)$ bound required by any algorithm maintaining an approximate matching, recent research has begun to focus on estimating the size of the maximum matching. In one of the few non-trivial graph streaming results using polylog n space, Kapralov et al. [210] obtained a polylogarithmic approximate estimate for randomly ordered streams. The remaining algorithms in this line of research focus on approximating matching sizes in classes of sparse graphs including planar graphs. The arboricity of a graph G is defined as $\max_{U\subseteq V}\left\lceil \frac{|E(U)|}{|U|-1}\right\rceil$. The number of high degree nodes and the number of edges that are not incident to high degree nodes are two quantities that relate the arboricity to the matching size. Indeed, McGregor and Vorotnikova showed that for graphs with arboricity a, $\sum_{(u,v)\in E}\min\left(\frac{1}{deg(u)},\frac{1}{deg(v)},\frac{1}{a+1}\right)$ determines the matching size up to a $(a+2)$ factor. Estimations of quantities related to this sum have then been used to obtain approximations to matching sizes. The first paper to consider matching approximation for these graphs was in the context of distributed computing [119]. Similar techniques were used by Esfandiari et al. [139] to obtain estimations in a streaming setting. Specifically for constant arboricity, they obtain a constant factor estimation using $\tilde{O}(n^{2/3})$ space in a single pass and $\tilde{O}(\sqrt{n})$ space using two passes or assuming randomly ordered streams. The authors also gave a lower bound of $\Omega(\sqrt{n})$ for any approximation better than $\frac{3}{2}$. This was subsequently extended to a $\tilde{O}(n^{4/5})$ in dynamic streams by Chitnis et al.[100] and Bury and the author. The work by Bury and the author [79] showed that an α-approximation of the unweighted matching size can be used to obtain an $O(\alpha^4)$-approximation for weighted matching. This was later improved to a $2\cdot\alpha$-approximation independently by McGregor and Vorotnikova [246], Grigorescu et al. [167] and by Bury and the author [79]. The related question of estimating the rank was also considered by Bury and the author [79]. The main result is that no insertion only streaming algorithm can maintain a $(1+O(t))$-approximation to the rank of a matrix using less than $\Omega(n^{1-1/t})$ space, which gives an exponential separation over estimating the number of non-zero entries of a vector [205]. This result was subsequently extended to other Shatten norms by Li and Woodruff [233]. Previously, all known publications on estimating matrix norms were for dynamic streams [21, 103, 231]. Recently, Assadi et al. [32] gave improved lower bounds for both insertion-only and turnstile streams. They showed that for any graph with arboricity α, obtaining an α-approximation in dynamic streams requires $\Omega(\sqrt{n}/\alpha^{2.5})$

space. For small approximation factors, they showed that $RS(n) \cdot n^{1-O(\varepsilon)}$ space is needed, where $RS(n)$ denotes the maximum number of edge disjoint matchings of size $\Theta(n)$ in a graph with n nodes, i.e. the maximum size of a $\Omega(1)$-Ruzsa-Szeméredi graph. Sharp bounds for $RS(n)$ are not known, but we know that $n^{1/\log\log n} \leq RS(n) \leq n/\log n$.

2.6.2 Point/Vertex Stream

Unlike the other streaming models, point streams are more loosely defined. The most commonly found assumption is that we are given a sequence of points from a metric space. The papers vary in the metrics considered. In finite metrics we are limited to storing points as whole (see the following definition). In Euclidean metrics, we may move the points, storing projections, grid corners, centroids, and other points not part of the input sequence. Moreover, even if two papers study, say, k-median in Euclidean spaces, one paper might measure the space in terms of the number of points stored, while another paper will measure the space in terms of bits stored. We will informally describe the models when reviewing the literature, but for the sake of this paper, we will constrain ourselves to the following.

Definition 2.6.4. Let (X, dist) be a metric space and let $A \subseteq X$.

- An *insertion-only point stream* is a sequence S of distinct elements $p \in X$ signifying that p is added to A.
- A *sliding window point stream* with window size N is a sequence S of distinct elements of X such that only the most recent N elements are contained in A.
- A streaming algorithm maintains a subset $P \subseteq A$ of points. Upon insertion of a point p, it is initially added to P. At any given time, the algorithm may irrevocably discard a point from P. For any two points $p, q \in P$, the algorithm may query $\text{dist}(p, q)$. Space is measured in terms of the size of $|P|$. A sliding window algorithm can query the remaining time to live (TTL) of any point $p \in P$.

It is difficult to assess the first point streaming algorithm. Certainly, space constrained algorithms have been published as far back as the late 60s, see MacQueen [236]. However, practical clustering algorithms designed with scalability issues in mind first appeared in the late 90s [67, 176, 289]. In 2000, Guha, Mishra, Motwani and O'Callaghan [174] published the first clustering paper with an explicit emphasis on streaming with guarantees on approximation ratio and space requirement, though Charikar et al. [91] had published a paper on incremental clustering for k-center even earlier in 1997.

Most theoretical papers assume that the points lie in Euclidean space and measure the space requirement of the algorithm in terms of used points. Clustering, specifically k-median, k-means, and k-center is probably the most studied problem, but a large number of papers also study other geometric problems, see for instance [3, 20, 87, 88, 150].

There exist two general (but by no means exclusive) approaches to solve k-median and k-means clustering in insertion-only streams. The first is to produce a clustering on the fly, see [14, 93, 174, 269] for results on k-means and the related k-median objective. The state of the art seems to be an algorithm by Braverman et al. [70] which produces a weighted set of $O(k \log n)$ points with constant approximation to the cost of an optimal k-means or k-median clustering.

The second approach is to aggregate the data for subsequent computation on the summary. Braverman et al. [70] augment their construction to provide good approximations for data sets satisfying a separation condition introduced by Ostrovsky et al. [257]. For general inputs, research has focused on constructions of coresets. Perhaps surprisingly, there exist coresets for k-means and k-median whose sizes have no dependency on the number of input points n [184, 226, 145]. Recently, Feldman et al. [147] and Cohen et al. [109] further proved that there exist coresets whose size does not depend on the dimension d. Whether this is also possible for k-median is an open question. For specific space bounds of various coreset constructions, we refer to Table 2.6.2. Coresets for k-means have the very useful property of being closed under union, that is, for two point sets, the union of coresets for both point sets is a coreset for the entire point set. This property allows us to transform an arbitrary offline coreset construction into a coreset for data streams via the merge and reduce framework introduced by Bentley and Saxe [60] by partitioning the input point set into a batch of points of small size (say $O(\log n)$), computing a coreset on each batch, and merging coresets bottom-up in a binary tree by recomputing a coreset of two coresets of equal depth. This framework does not come without a cost. The merging step incurs a loss in quality, that is, a coreset of two ϵ-coresets is a $2\epsilon + \epsilon^2$ coreset. Since the merge and reduce tree has depth $\log(n/\log n) \in O(\log n)$, this procedure introduces a dependency of a factor of $\log n$ on ϵ. In addition, we require the storage of at most one coreset at every level of the merge and reduce tree, incurring another $\log n$ in the space requirement. Finally, all known constructions for high dimensions fail with an adjustable probability δ. To limit the overall failure probability when processing a stream, δ is rescaled by the number of batches, incurring another factor of $O(\log n)$. The best dependency on $\log n$ is due to Langberg and Schulman [226] whose construction requires $\log^4 n$. There exist constructions not relying on the merge and reduce framework but processing each point online. While they typically have a better dependency on $\log n$, the best result by Fichtenberger et al. [149] is nevertheless exponential in the dimension. To the best of our knowledge, there exist no non-trivial lower bounds for k-means or k-median in insertion-only streams. Braverman et al. [68, 69] made some recent progress on adapting coresets for sliding windows. To the best of our knowledge, the only other sliding window clustering algorithm is due to Babcock et al. [38].

In dynamic geometric streams, the importance of precisely defining the input becomes more apparent. Here, the points are generally assumed to lie in discrete Euclidean space,

Algorithm	Offline Memory	Streaming Memory
Low Dimensions		
[185]	$O(k\epsilon^{-d}\log n)$	$O(k\epsilon^{-(d+1)}\log^{2d+2} n)$
[184]	$O(k^3\epsilon^{-(d+1)})$	$O(k^3\epsilon^{-(d+1)}\log^{d+2} n)$
[151]	$O(k\epsilon^{-d}\log n)$	$O(k\epsilon^{-(d+2)}\log^4 n)$
[149]	$O(k\epsilon^{-(d+2)}\log n)$	$O(k\epsilon^{-(d+2)}\log n)$
High Dimensions		
[95]	$O(d^2k^2\epsilon^{-2}\log^5 n)$	$O(d^2k^2\epsilon^{-2}\log^9 n)$
[146]	$O(k^2\epsilon^{-5})$	$O(k^2\epsilon^{-5}\log^7 n)$
[226]	$O(d^2k^3\epsilon^{-2})$	$O(d^2k^3\epsilon^{-2}\log^4 n)$
[145]	$O(dk\epsilon^{-4})$	$O(dk\epsilon^{-4}\log^6 n)$

Table 2.1: Comparison of memory demands, where polylogarithmic factors are suppressed and the memory to store a d-dimensional point is not specified. The constructions for high dimensions do not treat d as a constant and succeed with constant probability. [146] produces a weak coreset from which an $(1+\epsilon)$-approximation can be recovered. Any offline dependency on d may be replaced by k/ϵ via Theorem 7 of Cohen et al. [109] (see also the earlier work by Feldman et al. [147]) and any streaming dependency on d may be replaced by $k\epsilon^{-2}$ via Theorem 12 of Cohen et al. [109].

i.e. the space is covered by some finite grid of known granularity and the points can be placed on the corners of the grid. The merge and reduce framework is not easily applied in this setting and algorithms here utilize linear sketches such as quadtree decompositions or pyramid sketches [42, 151, 195, 197].

The k-center problem requires different techniques. For arbitrary metrics, the aforementioned paper by Charikar et al. [91] gave a constant approximation ratio which was further improved independently by McCutchen and Khuller [242] and Guha [170] to a $(2+\varepsilon)$-approximate algorithm using $O(k/\varepsilon \log 1/\varepsilon)$ space. Their algorithm is based on doubling techniques that we will also encounter in Chapter 5. Guha also gave an almost tight lower bound of $\Omega(n)$ space for streams of length n for any algorithm achieving a better approximation ratio than 2 and also gave a lower bound of $\Omega(k^2)$ points for any algorithm approximating the objective value beyond a factor of $2+1/k$. Further improvements are possible in Euclidean spaces. Zarrabi-Zadeh showed how to maintain coresets in streams using $O(k\varepsilon^{-d})$ points for k-center [286], yielding a $(1+\varepsilon)$-approximation. For small values of k, Kim and Ahn [217] were able to break the 2 barrier without having an exponential dependency on d, giving a $(1.8+\varepsilon)$-approximation while storing $O(2^k(k+3)!\varepsilon^{-1})$ points.

The 1-center problem in Euclidean space has also received a lot of attention. In their influential paper, Agarwal, Har-Peled and Varadarajan [3] proposed streaming algorithms for various extent problems in Euclidean spaces, including a $(1+\varepsilon)$ approximation for the minimum enclosing ball problem. The approaches were refined in subsequent papers [6, 85, 86, 287] with the currently best algorithm by Arya and Chan storing $O(\varepsilon^{-(d-1)/2})$ points [29]. Agarwal and Sharathkumar [5] showed that no algorithm with polynomial dependency on d can achieve a better approximation ratio than $(1+\sqrt{2})/2 \approx 1.207$. They also proposed an algorithm based on the gradient descent of Badoiu and Clarkson [43] which, after a re-analysis by Chan and Pathak [88], is now known to give a 1.22-approximation storing only a constant number of points. To our knowledge, there exists no work on sliding window algorithms for k-center or the minimum enclosing ball problem, though two papers deal with the related question of approximating the diameter in discrete Euclidean spaces [89, 143]. For dynamic streams in discrete Euclidean spaces, Andoni and Nguyen [20] proposed a data structure that stores the directional width in any direction. This data structure could conceivably be used to compute the minimum enclosing ball, though at the time it was not clear how to do this with feasible running times. This issue has been recently addressed by Chan [87].

3 Estimation of Matching Sizes

In this chapter we give some results on estimating matching sizes. For our upper bounds, we focus on dynamic streams, i.e. graph streams processing an arbitrary number of insertions and deletions. Our lower bound holds for adversarially ordered insertion-only streams.

3.1 Unweighted Matching Estimation for Dynamic Streams

We give two estimation algorithms for the size of a maximum matching. First, we see that it is easy to estimate the matching size in trees. Second, we show how to decide whether there exists a matching of size at least k and extend the result from [139] where the matching size of so called bounded arboricity graphs in insertion-only streams is extended to dynamic graph streams.

Estimating the Matching Size of Trees: Let $T = (V, E)$ be a tree with at least 3 nodes and let h_T be the number of internal nodes, i. e., nodes with degree greater than 1. It is easy to see that the matching size is bounded by h_T since every edge in a matching has to be incident to at least 1 internal node. A maximum matching is also lower bounded by $h_T/2$ which is a consequence of Hall's Theorem. Define the neighborhood $N(S)$ of a set of nodes S as the set of nodes adjacent to some node of S.

Theorem 3.1.1 (Hall [180])**.** *Let $G = (A, B, E)$ be a bipartite graph. G contains a matching of size $|A|$ if and only if $|N(S)| \geq |S|$ for all $S \subseteq A$.*

Using Hall's theorem, we can easily bound the size of maximum matchings in trees with respect to the number of inner nodes, i. e., the nodes with degree greater than 1.

Corollary 3.1.2. *Let $T = (V, E)$ be a tree and h_T be the number of nodes with degree greater than 1. Then the size of a maximum matching M^* is bounded by*

$$\frac{h_T}{2} \leq |M^*| \leq \max\{h_T, 1\}.$$

In order to estimate the matching size, we maintain an ℓ_0-estimator for the degree vector $d \in \mathbb{R}^n$ such that $d_i = deg(v_i) - 1$ holds at the end of the stream and with it $\ell_0(d) = h_T$. In other words, we initialize the vector by adding -1 to each entry and update the two corresponding entries when we get an edge deletion or insertion. Using Theorem 10 from [205] we can maintain the ℓ_0-Estimator for d in $O(\varepsilon^{-2} \log^2 n)$ space.

Theorem 3.1.3. *Let $T = (V, E)$ be a tree with at least 3 nodes and let $\varepsilon \in (0, 1)$. Then there is an algorithm that estimates the size of a maximum matching in T within a $(2 + \varepsilon)$-factor in the dynamic streaming model with high probability using 1-pass over the data and $O(\varepsilon^{-2} \log^2 n)$ space.*

Tutte Matrix based Estimation for Arbitrary Graphs: We now detail an algorithm determining the exact matching size up to a parameter k using roughly k^2 space based on the Tutte matrix[1]. Our aim is to randomly choose entries of a Tutte matrix and update this matrix with the corresponding value whenever an edge is inserted or deleted.

One crucial ingredient is the following result due to Clarkson and Woodruff [103], see Sarlos [267] for similar, slightly weaker statements.

Lemma 3.1.4 (Lemma 3.4 of [103])**.** *Given integer k and $\varepsilon, \delta > 0$, there is $m = O(k \log(1/\delta)/\varepsilon)$ and an absolute constant η such that if S is an $n \times m$ sign matrix with $\eta(k + \log(1/\delta))$- wise independent entries, then for an $n \times k$ matrix U with orthonormal columns, with probability at least $1 - \delta$, the spectral norm $||U^T S S^T U - U^T U||_2 \leq \varepsilon$.*

Since U is orthogonal, all singular values are 1. If we choose ε to be some constant, the singular values of $S^T U$ and U differ only by multiplicative constant factors close to 1, which also implies that $S^T U$ and U have the same rank. For our purposes, $\varepsilon = 1/3$ will be sufficient.

Corollary 3.1.5. *Given integer k and $\delta > 0$, there is $m = O(k \log(1/\delta))$ and an absolute constant η such that if S is an $n \times m$ sign matrix with $\eta(k + \log(1/\delta))$- wise independent entries, then for an $n \times k$ matrix U with orthonormal columns, with probability at least $1 - \delta$, the rank of $S^T U$ is identical to rank of U.*

Our algorithm now proceeds as follows, see also Algorithm 1. We initialize the Tutte matrix T of the input graph G with randomly chosen entries drawn from a k^2-independent hash function h assigning each edge a random value in $[O(k^2)]$. We then independently sample two sign matrices S_1 and S_2 where S_1, S_2 satisfying the conditions of Corollary 3.1.5. We maintain $S_1 T S_2$ now as follows. Whenever we process an operation on the edge (u, v), the appropriate random value of the corresponding entry in T is queried via h. This value is inserted into an $n \times n$ matrix H containing only 0 except for $H(u, v) = h(u, v)$ and $H(u, v) = -h(u, v)$ if (u, v) is inserted and $H(u, v) = -h(u, v)$ and $H(u, v) = h(u, v)$ if (u, v) is deleted. $S_1 T S_2$ can then be updated by adding $S_1 T S_2 + S_1 H S_2$. Note that we do not have to construct the entire matrix H. The correctness of this algorithm is an almost direct application of Corollary 3.1.5:

Theorem 3.1.6. *Let $G(V, E)$ be an arbitrary graph. Then there exists a dynamic streaming algorithm that either (1) outputs k if if the maximum matching is greater than k or (2) outputs maximum matching size. The algorithm uses $O(k^2 \log n)$ space and succeeds with constant probability.*

[1] We note that a sampling strategy from [100] could replace the Tutte matrix based estimation. In fact their result is somewhat stronger, as they show that using only slightly more space, they can recover any matching up to size k. Nevertheless, we believe that our technique may be of independent interest.

Algorithm 1 Tutte Matrix Streaming Estimation

Require: Graph $G(V, E)$, Stream S, integer $k > 0$
Ensure: $\min(k, \text{Matching Size of } G)$

Let $h : [n^2] \to [O(k^2)]$ be a k^2-wise independent hash function.
Let S_1 and S_2 be independent $n \times m$ sign matrices with $m = O(k)$ and $O(k)$ independent entries.
$M \in \mathbb{R}^{m \times m}$ initially with entries 0.
$H \in \mathbb{R}^{n \times n}$ initially with entries 0.
for all $(u, v, t) \in S$ **do**
  $H(u, v) \leftarrow (-1)^{1+t} h(n \cdot u + v)$
  $H(v, u) \leftarrow -(-1)^{1+t} h(n \cdot u + v)$
  $M \leftarrow S_1^T H S_2$
  $H \leftarrow 0$
end for
return $\text{rank}(M)$

Before we prove this theorem, we first remark that Theorem 2.4.3 still holds when we have limited independence.

Corollary 3.1.7. *Let $G = (V, E)$ be a graph and $r \in \mathbb{R}^{|E|}$ be a random vector with k^2-wise independent entries where each coordinate is uniformly chosen from $\{1, \ldots, ck^2\}$. Then we have*

$$rank(T_G(r)) = \min\left(k, \max_w \{rank(T_G(w))\}\right)$$

with probability at least $1 - 1/c$.

Proof. Consider a k by k sub-matrix T' of T with maximum possible rank. If $\max_w\{rank(T_G(w))\} \leq k$, then $\max_w\{rank(T_G(w))\} = \max_w\{rank(T'_G(w))\}$, otherwise $\max_w\{rank(T_G(w))\} = k$. The corollary now follows by applying Theorem 2.4.4 on T'. □

Proof of Theorem 3.1.6. We first argue correctness, then space. We randomly chose the weights of the Tutte matrix T from 1 to $4k^2$ such that the weights are k^2-wise independent (line 1 of Algorithm 1). By Corollary 3.1.7, Theorem 2.4.3 holds when we query the size of the matching with constant probability. It is straightforward to maintain $S_1^T T S_2$ whenever we receive and edge insertion or deletion (lines 6-11 of Algorithm 1).

What remains is to analyze the rank of $S_1^T T S_2$. First, let $r \leq k$ be the rank of T. Let $U_1 \Sigma U_2^T$ be the singular value truncated decomposition of T such that $U_1, U_2 \in \mathbb{R}^{n \times r}$ are orthogonal and $\Sigma \in \mathbb{R}^{r \times r}$ is diagonal with non-zero entries. Corollary 3.1.5 guarantees us that any rank up to k of $S_1^T U_1$ and $U_2^T S_2$ is preserved with constant probability. Since Σ is a diagonal matrix with non-zero entries and U_1 is orthogonal, $\text{rank}(U_1 \Sigma U_2^T S_2) = \text{rank}(U_1^T U_1 \Sigma U_2^T S_2) = \text{rank}(\Sigma U_2^T S_2) = \text{rank}(U_2^T S_2) = \text{rank}(U_2) = r$. By the same argument and independence of S_1 and S_2, $\text{rank}(S_1^T T S_2) = r$.

If $r > k$, we can decompose U_1 (and analogously U_2) into the sum of two orthogonal matrices U_k and U_R, where U_k consists of the first k columns of U_1 and U_R consists of the remaining columns of U_1. We apply the same line of reasoning as above onto U_k and note that the rank cannot decrease by adding $S^T U_R$.

The space bound of each $S_1^T T S_2$ is in $O(k^2 \log n)$ due to the dimension of the sign matrices via Corollary 3.1.5 and by observing that the magnitude of entries of $S_1^T T S_2$ is polynomial in n. Using Theorem 2.1.2 (Theorem 1.1 by Pagh and Pagh [258]), we can store h using $O(k^2 \log k)$ bits and S_1 and S_2 using $O(k)$ bits. Thus, the total space is dominated by $O(k^2 \log n)$. □

Theorem 3.1.6 can be used as a subroutine for approximating matching sizes in sparse graphs. We briefly outline the results in this line of research. The arboricity $a(G)$ of G is a kind of density measure: The number of edges in every induced subgraph of size s in G is bounded by $s \cdot a(G)$. Formally, the arboricity $a(G)$ of G is defined by $a(G) = \max_{U \subseteq V} \left\lceil \frac{|E(U)|}{|U|-1} \right\rceil$.

Since estimating the matching size is often difficult, researchers have instead focused on simpler estimation tasks whose errors are parameterized by the arboricity of the graph. In one way or the other, these estimations are based on approximating functions of the degrees of the nodes based on a sampled subgraph. The currently best estimator in this vein by McGregor and Vorotnikova [246] approximates the function $A := \sum_{(u,v) \in V} \min(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{a(G)+1})$, which approximates the matching up to a $(a(G)+2)$-factor.

Theorem 3.1.8. *Let match(G) be the size of the maximum cardinality matching in $G(V, E)$. For an edge $e = \{u, v\} \in E$ define $x_e = \min\left(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha+1}\right)$. Then,*

$$match(G) \leq (\alpha + 1) \sum_{e \in E} x_e \leq (\alpha + 2) match(G)$$

From Theorem 3.1.8, we know we can estimate the size of the maximum cardinality via the following quantity,

$$A := \sum_{\{u,v\} \in E} \min\left(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha+1}\right) .$$

A in turn may be estimated via the quantity,

$$A_S := \sum_{\{u,v\} \in E : u,v \in S} \min\left(\frac{1}{\deg(u)}, \frac{1}{\deg(v)}, \frac{1}{\alpha+1}\right) .$$

where S is a subset of V formed by sampling each node independently with probability p. McGregor and Vorotnikova [246] show that A_S is within a $1+\epsilon$ factor of Ap^2 with probability at least $3/4$ assuming p is sufficiently large.

Lemma 3.1.9. *If $p \geq \sqrt{12\epsilon^{-2}A^{-1}}$, then $\mathbb{P}[|A_S - Ap^2| \leq \epsilon \cdot Ap^2] \geq 3/4$.*

Algorithm 2 Approximation of Weighted Matching from [279]

Require: Graph $G = (V, E = \bigcup_{i=1}^{t} E_i)$
Ensure: Matching M
  $M \leftarrow \emptyset$
  for $i = t$ to 1 **do**
    Find a maximal matching M_i in $G_i = (V, E_i)$.
    Add M_i to M.
    Remove all edges e from E such that $e \in M_i$ or e shares a node with an edge in M_i.
  end for
  return M

We use Theorem 3.1.6 to determine the matching size up to $n^{2/5}$. For larger matchings, we apply Lemma 3.1.9 with $p = \Theta(\varepsilon^{-1}/n^{4/5})$.

Theorem 3.1.10. *There exists a single pass data stream algorithm using $\tilde{O}(\alpha\epsilon^{-1}n^{4/5}\log\delta^{-1})$ space that returns a $(\alpha+2)(1+\epsilon)$ approximation of the maximum matching with probability at least $1-\delta$.*

For related, earlier estimators, we refer to Esfandiari et al. [139] and Czygrinow et al.[119]

3.2 Weighted Matching

We start by describing the parallel algorithm by Uehara and Chen [279], see Algorithm 2. Let $\gamma > 1$ and $k > 0$ be constant. We partition the edge set by t ranks where all edges e in rank $i \in \{1, \ldots, t\}$ have a weight $w(e) \in \left(\gamma^{i-1} \cdot \frac{w_{max}}{kn}, \gamma^i \cdot \frac{w_{max}}{kn}\right]$ where w_{max} is the maximal weight in G. For simplicity, assume $\frac{w_{max}}{kn}$ to be scaled to 1. Let $G' = (V, E, w')$ be equal to G but each edge e in rank i has weight γ^i for all $i = 1, \ldots, t$. Starting with $i = t$, we compute an unweighted maximal matching M_i considering only edges in rank i (in G') and remove all edges incident to a matched node. Continue with $i-1$. The weight of the matching $M = \bigcup M_i$ is $w(M) = \sum_{i=1}^{t} \gamma^i \cdot |M_i|$ and satisfies $w_G(M^*) \geq w_{G'}(M) \geq \frac{1}{2\gamma} \cdot w_G(M^*)$ where M^* is an optimal weighted matching in G.

The previous algorithms [118, 141, 243, 137, 138, 288] for insertion-only streams use a similar partitioning of edge weights. Since these algorithms store no more than one maximal matching per rank, they cannot compute residual maximal matchings, but by charging the smaller edge weights into the higher ones the resulting approximation factor can be made reasonably close to that of Uehara and Chen.

In order to adapt this idea to our setting, we need to work out the key properties of the partitioning and how we can implement it in a stream. Recalling the partitioning of Uehara and Chen, we disregard all edges with weight smaller than $\frac{w_{max}}{kN}$ which is possible because the contribution of these edges is at most $\frac{N}{2} \cdot \frac{w_{max}}{kN} = \frac{w_{max}}{2k} \leq \frac{OPT}{2k}$ where OPT is the weight of an optimal weighted matching. Thus, we can only consider edges with larger weight and it

is also possible to partition the set of edges in a logarithmic number of sets. Here, we use the properties that edge weights within a single partition set are similar and that $1 \leq \frac{w(e)}{w(e')} \leq \gamma^2$ for two edges $e \in E_i$ and $e' \in E_{i-1}$ with $i \in \{2, \ldots, t\}$. These properties are sufficient to get a good approximation on the optimal weighted matching which we show in the next lemma. The proof is essentially the same as in [279].

Lemma 3.2.1. *Let $G = (V, E, w)$ be a weighted graph and $\varepsilon > 0$ be an approximation parameter. If a partitioning $E_1, \ldots, E_t$ of E and a weight function $w' : E \to \mathbb{R}$ satisfies*

$$\frac{1}{1+\varepsilon} \leq \frac{w'(e)}{w(e)} \leq 1 \text{ for all } e \in E \quad \text{and} \quad \frac{w(e_1)}{w(e_2)} \leq 1+\varepsilon \quad \text{and} \quad w(e) < w(e')$$

for all choices of edges $e_1, e_2 \in E_i$ and $e \in E_i, e' \in E_j$ with $i < j$ and $i, j \in \{1, \ldots, t\}$ then Algorithm 2 returns a matching $M = \bigcup_{i=1}^{t} M_i$ with

$$\frac{1}{2(1+\varepsilon)^2} \cdot w(M^*) \leq w'(M) \leq w(M^*)$$

where M^* is an optimal weighted matching in G.

Proof. The first property $\frac{1}{1+\varepsilon} \leq \frac{w'(e)}{w(e)} \leq 1$ for all $e \in E$ implies that $\frac{w(S)}{1+\varepsilon} \leq w'(S) \leq w(S)$ for every set of edges $S \subseteq E$. Thus, it remains to show that $\frac{1}{2(1+\varepsilon)} \cdot w(M^*) \leq w(M) \leq w(M^*)$. Since M^* is an optimal weighted matching, it is clear that $w(M) \leq w(M^*)$. For the lower bound, we distribute the weight of the edges from the optimal solution to edges in M. Let $e \in M^*$ and $i \in \{1, \ldots, t\}$ such that $e \in E_i$. We consider the following cases:

1. $e \in M_i$: We charge the weight $w(e)$ to the edge itself.
2. $e \notin M_i$ but at least one node incident to e is matched by an edge in M_i: Let $e' \in M_i$ be an edge sharing a node with e. Distribute the weight $w(e)$ to e'.
3. $e \notin M_i$ and there is no edge in M_i sharing a node with e: By Algorithm 2, there has to be an edge $e' \in M_j$ with $j > i$ which shares a node with e. We distribute the weight $w(e)$ to e'.

Since M^* is a matching, there can only be at most two edges from M^* distributing their weights to an edge in M. We know that $\frac{w(e)}{w(e')} \leq 1+\varepsilon$ for all choices of two edges $e, e' \in E_i$ with $i \in \{1, \ldots, t\}$ which means that in the case 2. we have $w(e) \leq (1+\varepsilon) \cdot w(e')$. In case 3. it holds $w(e) < w(e')$. Thus, the weight distributed to an edge e' in M is at most $2(1+\varepsilon)w(e')$. This implies that $w(M^*) = \sum_{e \in M^*} w(e) \leq \sum_{e' \in M} 2(1+\varepsilon) \cdot w(e') = 2(1+\varepsilon) \cdot w(M)$ which concludes the proof. □

Using Lemma 3.2.1, we can partition the edge set in a stream in an almost oblivious manner: Let $(e, w(e))$ be the first inserted edge. Then an edge e' belongs to E_i iff $(1+\varepsilon)^{i-1} \cdot w(e) <$

Algorithm 3 Estimation of Weighted Matching

Require: Graph $G = (V, E = \bigcup_{i=1}^{t} E_i)$, unweighted estimation routine A
Ensure: Estimated weight $\widehat{W}$
  for $i = t$ to 2 **do**
    Use A to estimate the size of a maximum matching in $G_i = (V, \bigcup_{j=i}^{t} E_j)$.
    $\widehat{W} \leftarrow \widehat{W} + (w'(i) - w'(i-1)) \cdot A(G_i)$.
  end for
  Use A to estimate the size of a maximum matching $G_1 = (V, \bigcup_{j=1}^{t} E_j)$.
  $\widehat{W} \leftarrow \widehat{W} + A(G_1)$.
  return $\widehat{W}$

$w(e') \leq (1+\varepsilon)^i \cdot w(e)$ for some $i \in \mathbb{Z}$. For the sake of simplicity, we assume that the edge weights are in $[1, W]$. Then the number of sets is $O(\log W)$. We would typically expect $W \in \operatorname{poly} n$ as otherwise storing weights becomes infeasible. In the following, denote by $w'(i)$ the weight of edges in rank E_i. We now are able to give our weighted estimation algorithm and state our main theorem.

Theorem 3.2.2. *Let $G = (V, E, w)$ be a weighted graph where the weights are from $[1, W]$. Let A be an algorithm that returns a λ-estimator $\widehat{M}$ for the size of an unweighted maximum matching M of a graph with $1/\lambda \cdot |M| \leq \widehat{M} \leq |M|$ with failure probability at most δ and needs space S. If we partition the edge set into sets $E_1, \ldots, E_t$ with $t = \lfloor \log_{1+\varepsilon} W \rfloor \in O(\varepsilon^{-1} \log W)$ where E_i consists of all edges with weight in $[(1+\varepsilon)^{i-1}, (1+\varepsilon)^i)$, and use A as the unweighted matching estimator in Algorithm 3 with $w'(i) = (1+\varepsilon)^{i-1}$, then the algorithm returns a $2 \cdot (1+\varepsilon)^2\lambda$-estimator $\widehat{W}$ for the weight of the maximum weighted matching with failure probability at most $\delta \cdot (t+1)$ using $O(S \cdot t)$ space., i.e. $\frac{1}{2(1+\varepsilon)^2\lambda} \cdot w(M^*) \leq \widehat{W} \leq w(M^*)$ where M^* is an optimal weighted matching.*

Proof. In the following denote by $\operatorname{match}(G_i)$ the size of the maximum matching in the graph $G_i = (V, , \bigcup_{j=i}^{t} E_j)$. Let M' be the matching computed by Algorithm 2 and denote by $S_i := M' \cap \left(\bigcup_{j \geq i}^{t} E_j\right)$ the partial matching of $G\left(V, \left(\bigcup_{j \geq i}^{t} E_j\right)\right)$ for any rank $i \in \{1, \ldots t\}$. Note that $S_1 := M$. We can then decompose the objective function as follows

$$\begin{aligned} w(M) &\geq \sum_{i=1}^{t} \sum_{e \in M \cap E_i} w(e) = \sum_{i=1}^{t} \sum_{e \in M \cap E_i} (1+\varepsilon)^{i-1} \\ &= \sum_{i=2}^{t} |S_i| \cdot \left((1+\varepsilon)^{i-1} - (1+\varepsilon)^{i-2}\right) + |S_1|. \end{aligned}$$

Assume now that each call to the unweighted λ-estimation algorithm for the maximum match-

ing of $G\left(V, \left(\bigcup_{j\geq i}^{t} E_j\right)\right)$ succeeds, which happens with probability $1-\delta(t+1)$. We have

$$\begin{aligned}
\widehat{W} &= \sum_{i=2}^{t} A(G_i)\cdot\left((1+\varepsilon)^{i-1}-(1+\varepsilon)^{i-2}\right)+A(G_1)\\
&\geq \sum_{i=2}^{t}\frac{1}{\lambda}\cdot\mathrm{match}(G_i)\left((1+\varepsilon)^{i-1}-(1+\varepsilon)^{i-2}\right)+\frac{1}{\lambda}\cdot\mathrm{match}(G_1)\\
&\geq \sum_{i=2}^{t}\frac{1}{\lambda}\cdot|M^*\cap E_i|\left((1+\varepsilon)^{i-1}-(1+\varepsilon)^{i-2}\right)+\frac{1}{\lambda}\cdot|M^*\cap E_1|\geq\frac{1}{\lambda(1+\varepsilon)^2}\cdot w(M^*)
\end{aligned}$$

Since S_i is maximal w.r.t. $G\left(V, \left(\bigcup_{j\geq i}^{t} E_j\right)\right)$, we have $A(G_i)\leq 2|S_i|$. Then for the upper bound

$$\begin{aligned}
\widehat{W} &= \sum_{i=2}^{t} A(G_i)\cdot\left((1+\varepsilon)^{i-1}-(1+\varepsilon)^{i-2}\right)+A(G_1)\\
&\leq 2\cdot\left(\sum_{i=2}^{t}|S_i|\cdot\left((1+\varepsilon)^{i-1}-(1+\varepsilon)^{i-2}\right)+|S_1|\right)\leq 2w(M)\leq 2w(M^*).
\end{aligned}$$

Combining these two bounds and rescaling $\widehat{W}$, the theorem follows. □

3.3 Lower Bounds for Insertion-Only Streams

Esfandiari et al. [139] showed a space lower bound of $\Omega(\sqrt{n})$ for any estimation better than 3/2. Their reduction uses the Boolean Hidden Matching Problem introduced by Bar-Yossef et al. [53], and further studied by Gavinsky et al. [159]. We will use the following generalization due to Verbin and Yu [282]. We first require a bit of notation. A t-hypergraph $G(V,E)$ is a set of nodes V and a set of hyperedges E, where each hyperedge e is a subset of exactly t nodes of V.

Definition 3.3.1 (Boolean Hidden Hypermatching Problem [282])**.** In the *Boolean Hidden Hypermatching* Problem $BHH_{t,n}$ Alice gets a vector $x\in\{0,1\}^n$ with $n=2kt$ and $k\in\mathbb{N}$. Bob gets a hypergraph with n nodes with some arbitrary but fixed ordering and a perfect t-hypermatching M, i. e., each hyperedge has exactly t coordinates and each node is contained in exactly one hyperedge, and a string $w\in\{0,1\}^{n/t}$. We denote the vector of length n/t given by $(\bigoplus_{1\leq i\leq t} x_{M_{1,i}},\ldots,\bigoplus_{1\leq i\leq t} x_{M_{n/t,i}})$ by Mx where $(M_{1,1},\ldots,M_{1,t}),\ldots,(M_{n/t,1},\ldots,M_{n/t,t})$ are the edges of M. The problem is to return 1 if $Mx\oplus w=1^{n/t}$ and 0 if $Mx\oplus w=0^{n/t}$, otherwise the algorithm may answer arbitrarily.

Verbin and Yu [282] showed a lower bound of $\Omega(n^{1-1/t})$ for the randomized one-way communication complexity for $BHH_{t,n}$. For our reduction we require $w=0^{n/t}$ and $x\in\{0,1\}^n$ has exactly $n/2$ bits set to 1. We denote this problem by $BHH^0_{t,n}$. We can show that this does not reduce the communication complexity.

Lemma 3.3.2. *Let n be a multiple of $4t$. The communication complexity of $BHH^0_{t,4n}$ is lower bounded by the communication complexity of $BHH_{t,n}$.*

Proof. Alice is given a boolean vector $x \in \{0,1\}^n$ with $n = 4kt$ for some $k \in \mathbb{N}$ and Bob a t-hypermatching on n nodes with some arbitrary but fixed ordering and a boolean vector $w \in \{0,1\}^{n/t}$. From their respective inputs, Alice will construct a boolean vector $x' \in \{0,1\}^{4n}$ and Bob will construct a t-hypermatching M' on $4n$ nodes such that $M'x' \oplus 0 = 0$ if $Mx \oplus w = 0$ and $M'x' \oplus 0 = 1$ if $Mx \oplus w = 1$ described as follows.

First, let us assume that t is odd. Alice constructs $x' = [x^T x^T \overline{x}^T \overline{x}^T]^T$ as the concatenation of two identical copies of x and two identical copies of the vector resulting from the bitwise negation of x. Without loss of generality, let $\{y_1, \ldots, y_t\} \in M$ be the l-th hyperedge of M, where y_i denotes the ith node of the hypergraph in the arbitrary but fixed ordering. Bob adds the following four hyperedges to M':

- $\{x_1, \overline{x_2}, \ldots, \overline{x_t}\}, \{\overline{x_1}, x_2, \overline{x_3}, \ldots, \overline{x_t}\}$, $\{\overline{x_1}, \overline{x_2}, x_3, \ldots, x_t\}$, and $\{x_1, \ldots, x_t\}$ if $w_l = 0$,
- $\{\overline{x_1}, x_2, \ldots, x_t\}, \{x_1, \overline{x_2}, \ldots, x_t\}$, $\{x_1, x_2, \overline{x_3}, \ldots, \overline{x_t}\}$, and $\{\overline{x_1}, \ldots, \overline{x_t}\}$ if $w_l = 1$.

The important observation here is that, since t is odd, we flip even number of bits in the case $w_l = 0$ and an odd number of bits if $w_l = 1$. Since every bit flip results in a change of the parity of the set of bits, the parity does not change iff we flip an even number of bits. Therefore, $w_l \oplus x_1 \oplus \ldots \oplus x_t = 0$ iff the parity of each of the corresponding new hyperedges is 0. Applying the same reasoning to all hyperedges, we deduce that $M'x' = 0^{4n/t}$ if $Mx \oplus w = 0^{n/t}$ and $M'x' = 1^{4n/t}$ if $Mx \oplus w = 1^{n/t}$. The number of ones in $x' \in \{0,1\}^{4n}$ is exactly $2n$. If t is even, we can just change the cases for the added edges such that we flip an even number of bits in the case $w_l = 0$ and an odd number of bits if $w_l = 1$. Overall, this shows that a lower bound for $BHH_{t,n}$ implies a lower bound for $BHH^0_{t,4n}$. □

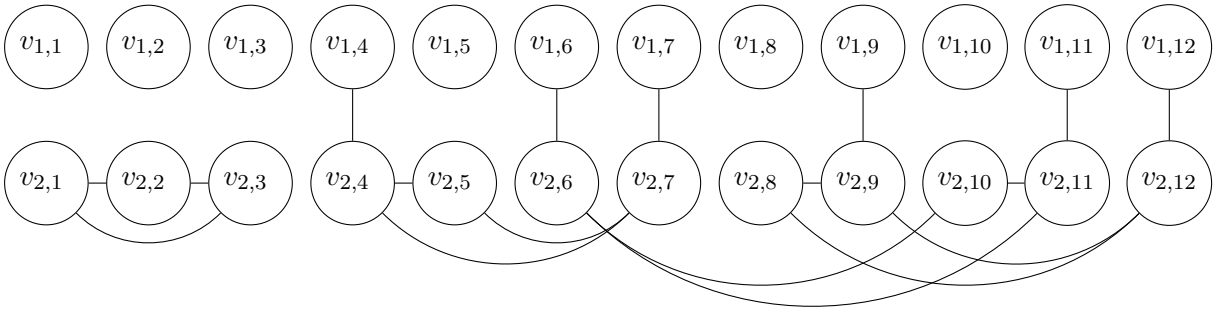

Figure 3.1: Worst case instance for $t = 3$. Bob's hypermatching corresponds to disjoint 3-cliques among the lower nodes and Alice' input vector corresponds to the edges between upper and lower nodes.

Theorem 3.3.3. *Any randomized streaming algorithm that approximates the maximum matching size within a* $1 + \frac{1}{3t/2-1} - \varepsilon$ *factor for* $t \geq 2$ *and any* $\varepsilon > 0$ *needs* $\Omega(n^{1-1/t})$ *space.*

Proof. Let $n = 4kt$ for some integer k. Alice is given boolean vector $x \in \{0,1\}^n$ with exactly $n/2$ indexes set to 0 and Bob is given a perfect t-hypermatching on a graph with n nodes. It is promised that either $Mx = 0^{n/t}$ or $Mx = 1^{n/t}$. Both players add edges to a graph G containing $2n$ nodes based on their respective inputs. For each index x_i we have two nodes $v_{1,i}, v_{2,i}$ and Alice adds the edge $\{v_{1,i}, v_{2,i}\}$ iff $x_i = 1$. For each edge $\{y_{i_1}, \ldots, y_{i_t}\} \in M$. Bob adds a t-clique consisting of the nodes $v_{2,i_1}, \ldots, v_{2,i_t}$. Alice now runs a streaming algorithm approximating the size of the maximum matching and sends the memory of the streaming algorithm to Bob. Bob then computes a $1 + \frac{1}{3t/2-1} - \varepsilon$ estimation of the maximum matching size of G. In the following we show that this approximation is sufficient to distinguish between the cases $Mx = 0^{n(t}$ or $Mx = 1^{n/t}$. This in turn shows that the memory of the message sent by Alice, i.e. the space requirement of the streaming algorithm is lower bounded by $BHH^0_{t,n}$.

We first consider the case where t is odd. We know that the maximum matching of G is at least $n/2$ because x has exactly $n/2$ ones. Since Bob adds a clique $v_{2,i_1}, \ldots, v_{2,i_t}$ for every hyperedge $\{y_{i_1}, \ldots, y_{i_t}\} \in M$ it is always possible to match all (or all but one) nodes $v_{2,i}$ of the clique whose corresponding bit is 0. In the case of $Mx = 0^{n/t}$ the parity of every edge is 0, i. e., the number of nodes whose corresponding bit is 1 is even. Let $M_{2i} \subseteq M$ be the hyperedges containing exactly $2i$ one bits and define $l_{2i} := |M_{2i}|$. Then we know $n/2 = \sum_{i=0}^{\lfloor t/2 \rfloor} 2i \cdot l_{2i}$ and $|M| = n/t = \sum_{i=0}^{\lfloor t/2 \rfloor} l_{2i}$. For every hyperedge in M_{2i} the size of the maximum matching within the corresponding subgraph of G is exactly $2i + \lfloor (t-2i)/2 \rfloor = 2i + \lfloor t/2 \rfloor - i$ for every $i = 0, \ldots, \lfloor t/2 \rfloor$ (see Fig. 3.1). Thus, we have a matching of size

$$\sum_{i=0}^{\lfloor t/2 \rfloor} (2i + (\lfloor t/2 \rfloor - i)) l_{2i} = \frac{n}{2} + \frac{t-1}{2} \cdot \frac{n}{t} - \frac{n}{4} = \frac{3n}{4} - \frac{n}{2t}.$$

If we have $Mx = 1^{n/t}$ then let $M_{2i+1} \subseteq M$ be the hyperedges containing exactly $2i+1$ one bits and define $l_{2i+1} := |M_{2i+1}|$. Again, we know $n/2 = \sum_{i=0}^{\lfloor t/2 \rfloor} (2i+1) \cdot l_{2i+1}$ and $|M| = n/t = \sum_{i=0}^{\lfloor t/2 \rfloor} l_{2i+1}$. For every edge in M_{2i+1} the size of the maximum matching within the corresponding subgraph is exactly $2i + 1 + (t - 2i - 1)/2 = 2i + 1 + \lfloor t/2 \rfloor - i$ for every $i = 0, \ldots, \lfloor t/2 \rfloor$. Thus, the maximum matching has a size

$$\sum_{i=0}^{\lfloor t/2 \rfloor} (2i + 1 + (\lfloor t/2 \rfloor - i)) l_{2i+1} = \frac{n}{2} + \frac{t-1}{2} \cdot \frac{n}{t} - \frac{1}{2} \sum_{i=0}^{\lfloor t/2 \rfloor} (2i+1) \cdot l_{2i+1} + \frac{n}{2t} = \frac{3n}{4}.$$

For t even, the size of the matching is

$$\sum_{i=0}^{t/2} (2i + (t-2i)/2) l_{2i} = \frac{n}{2} + \frac{t}{2} \cdot \frac{n}{t} - \frac{n}{4} = \frac{3n}{4}$$

if $Mx = 0^{n/t}$. Otherwise, we have

$$\begin{aligned}\sum_{i=0}^{t/2}\left(2i+1+\left\lfloor\frac{t-2i-1}{2}\right\rfloor\right)l_{2i+1} &= \frac{n}{2}+\sum_{i=0}^{t/2}(t/2-i-1)l_{2i+1}\\ &= \frac{n}{2}-(t/2-1)\cdot\frac{n}{t}-\frac{n}{4}+\frac{n}{2t}=\frac{3n}{4}-\frac{n}{2t}.\end{aligned}$$

As a consequence, every streaming algorithm that computes an α-approximation on the size of a maximum matching with

$$\alpha < \frac{(3/4)n}{((3/4)-1/(2t))n} = 1/(1-4/6t) = 1+\frac{1}{3t/2-1}$$

can distinguish between $Mx = 0^{n/t}$ and $Mx = 1^{n/t}$ and, thus, needs $\Omega(n^{1-1/t})$ space. □

Using the relationship between rank and Tutte-matrix established by Theorem 2.4.3 and 2.4.4, we can now prove the following corollary.

Corollary 3.3.4. *Any randomized streaming algorithm that approximates* $\operatorname{rank}(A)$ *of* $A \in \mathbb{R}^{n\times n}$ *within a* $1+\frac{1}{3t/2-1}-\varepsilon$ *factor for* $t \geq 2$ *and any* $\varepsilon > 0$ *requires* $\Omega(n^{1-1/t})$ *space.*

Proof. Given an instance of $BHH^0_{t,n}$, Alice and Bob construct the adjacency matrix as described in Theorem 3.3.3. They further choose each entry of the Tutte-matrix uniformly at random from $[n^2]$ from public randomness. Then approximating the rank of the Tutte-matrix within a factor $1+\frac{1}{3t/2-1}-\varepsilon$ approximates the matching within the same factor and solves $BHH^0_{t,n}$. □

4 Similarity Search in Dynamic Streams

Similarity measures between two bit-vectors are a basic building block for many data analysis tasks. In this chapter, we focus on the Jaccard similarity defined as $\frac{|A \cap B|}{|A \cup B|}$ for two item sets A and B, encountered in a wide range of applications such as clustering [175], plagiarism detection [76], association rule mining [106], collaborative filtering [120], recommender systems [40], and web compression [97]. We consider a scenario of nearest neighbor reporting, first described by Cohen et al. [106] Assume a collection of n item sets, where each item set contains items from a universe U. Our goal is to quickly find similar item sets with respect to the Jaccard index.

To report all pairs with similarity greater than some threshold T, we can evaluate the Jaccard index for all pairs. This straightforward approach requires $O(n^2)$ evaluations, even if the number of pairs whose similarity exceeds or is close to T, is far lower. Cohen et al. [106] presented a different approach with fewer evaluations. It's main idea is the use of a filtering mechanism to remove pairs of low similarity from consideration. There are two basic steps: first, we require an efficient way to store and compute similarities and second, we use this new similarity representation to quickly generate candidate pairs. For a more detailed overview on this approach see Section 4.1.

Cohen et al.[106] describe multiple ways to do this, given that the similarity measure admits a locality sensitive hashing scheme (LSH), see Definition 2.2.5 by Charikar [90]. Locality sensitive hashing for the Jaccard similarity (also known as min-hashing) has a long history with many papers analyzing properties of various families of hash functions and applying them to problems related to nearest neighbor searching. Despite the fact that min-hashing can immediately be implemented in an insertion-only environment and can also be extended to sliding window streams [125], no work exists on min-hashing when facing deletions. Deletions often occur in data changes over time and in particular in the context of recommender systems where users might significantly alter their profiles. Here, we initiate the study of maintaining locality sensitive hashing schemes in these dynamic streaming environments.

4.1 A Brief Survey on LSH-based Filtering (Cohen et al. [106])

In this section, we first give a high-level description of the ideas used by Cohen et al. [106] to quickly filter out low-similarity pairs. We then describe how this algorithm can be improved with space and streaming considerations in mind.

Succint Similarity representation For each item, we use a small representation known as a *fingerprint* or *min-hash*. Here, we randomly permute the elements of U and the hash value $h(A)$ of an item set A is the index of the first non-zero item of A. It is easy to verify that if the permutation is uniform, then $\mathbb{P}[h(A) = h(B)] = \frac{|A \cap B|}{|A \cup B|}$. Using multiple min-hashes, we gain a succinct representation of the original input matrix M defined in Figure 4.1. If the

$$\begin{array}{c} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix} \end{array} \longrightarrow \begin{array}{c} \begin{matrix} A & B & C & D \end{matrix} \\ \begin{pmatrix} 3 & 4 & 6 & 3 \\ 2 & 2 & 2 & 4 \\ 3 & 2 & 5 & 5 \end{pmatrix} \begin{matrix} h_1 \\ h_2 \\ h_3 \end{matrix} \end{array}$$

Figure 4.1: The figure gives an example for a min-hash based representation of a characteristic boolean matrix M (left matrix), where the columns are the item sets and the rows are the items,. In the right matrix M', the columns are still the item sets and the rows contain the fingerprints. The hash values of h_1 are given by the permutation $\Pi_1 = \{1 \to 4, 2 \to 5, 3 \to 1, 4 \to 3, 5 \to 6, 6 \to 2\}$; for h_2 by the permutation $\Pi_2 = \{1 \to 5, 2 \to 1, 3 \to 4, 4 \to 2, 5 \to 3, 6 \to 6\}$ and for h_3 by the permutation $\Pi_3 = \{1 \to 4, 2 \to 3, 3 \to 2, 4 \to 6, 5 \to 1, 6 \to 5\}$.

indexes are larger than T, we can use roughly $1/T$ fingerprints from independent random permutations to accurately estimate the similarity of two profiles (see also Theorem 1 of [106] for a precise statement).

Candidate Generation The candidate generation is composed of two basic steps: (1) tuning the LSH probabilities and (2) an efficient iteration over the hash-values

We can combine multiple fingerprints to amplify the probability of selecting high similar items and lower the probability in case of a small similarity: Let $p, q \in \mathbb{N}$. Then we generate p independent fingerprints and only add a pair to the output set iff all p hash values are equal. This procedure is repeated q times and the final output set contains all pairs which appear at least once in an output set of the q repetitions. The probability that a pair with similarity s is in the output set is $1 - (1 - s^p)^q$.

We now outline how to efficiently use the hash tables provided that the data set is well-behaved. The first is to use *row sorting*, see also Figure 4.1. Here for each row, we sort the columns increasing by their respective fingerprint values. We use n counters for the first column to count the number of agreements with the other columns and hence estimate their similarities. For the next column, we only reinitialize the counters that were incremented at least once. Our hope is that these pairs were indeed similar and that most counters will not have to be reinitialized. The time required for this algorithm is the sorting which for k hash

functions is $O(kn \log n)$ time. The number of counter initializations is equal to the number of increments. The expected number of increments for column A is exactly $\sum_{X \neq A \in N} S(A, X)$, hence the expected cost of all increments is $O(kn^2 \cdot \overline{S})$, where $\overline{S}$ is the average similarity. If $\overline{S}$ is small (i.e. most pairs have similarity near 0 and there are only few high similarity pairs), we can think of $\overline{S}n^2$ roughly corresponding to the size of the output.

$$\begin{array}{cccc} A & B & C & D \\ \begin{pmatrix} 3 & 4 & 6 & 3 \\ 2 & 2 & 2 & 4 \\ 3 & 2 & 5 & 5 \end{pmatrix} & \begin{matrix} h_1 \\ h_2 \\ h_3 \end{matrix} \end{array} \longrightarrow \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ \begin{pmatrix} - & - & \{A,D\} & \{B\} & - & \{C\} \\ - & \{A,B,C\} & - & \{D\} & - & - \\ - & \{B\} & \{A\} & - & \{C,D\} & \end{pmatrix} & \begin{matrix} h_1 \\ h_2 \\ h_2 \end{matrix} \end{array}$$

Figure 4.2: Having sorted each row increasingly, we initialize counters (A, B), (A, C), and (A, D). We obtain $S(A, B) = S(A, C) = S(A, D) = \frac{1}{3}$. Since all counters were used, we reinitialize (B, C) and (B, D), for which obtain $S(B, C) = \frac{1}{3}$ and $S(B, D) = 0$. Since (B, D) was not used, we can reuse the counter for (C, D), for which we obtain $S(C, D) = \frac{1}{3}$. Note that the entire domain (which may be large) is only sorted for illustrative purposes. Each row may in fact be only sorted w.r.t the key given by the left matrix.

Our second option is a *hash-count* approach. We associate a bucket for each min-hash value of each row and store an index of every item set hashed to the bucket. As was the case for the row-sorting algorithm, we iterate over all columns and maintain counters for the pairs, which are incremented if two item sets are hashed to the same bucket. The expected number of increments to a counter of the pair (A, B) can be shown to lie between $\min(k, |A \cup B|) \cdot S(A, B)$ and $\min(2k, |A \cup B|) \cdot S(A, B)$, so we again have an expected running time of $O(kn^2 \cdot \overline{S})$.

Space and Streaming Considerations The generation of the smaller representation matrix M' can be readily done for insertion-only streams. Whenever we process an update (i, j) signifying that item j has been added to item set i, we check for each row associated with the hash function h whether the associated $h(j)$ is smaller than the current fingerprint and update accordingly.

To make this algorithm more space-efficient, we have further options. A uniform random family of random permutations require an exponential amount of space to store and are infeasible, see Table 1 of Broder et al. [75] for exact space bounds for various classes of hash families. Thorup [274] showed that using extremely space efficient 2-wise independent hash functions also works. Specifically, if the similarity is large enough, we require only a constant number of fingerprints from 2-wise independent hash functions to achieve the same estimation guarantees as min-hashing. Though less relevant for this thesis, further papers have also considered the size of fingerprints. b-bit, introduced by Li and König [229] first computes a fingerprint for instance via min-hashing and then hashes the fingerprint down to

b bits. 2-bit hashing was proven to be space optimal by Pagh et al. [259].

The related question of maintaining small fingerprints in streams, was also investigated by Bachrach and Porat [41]. For any given ℓ_p vector norm, they observed that $\ell_p(a-b)^p = |A \triangle B| = |A \cup B| - |A \cap B|$, where a and b are the characteristic binary vectors of two item sets A and B, respectively. Provided that $\frac{|A \cap B|}{|A \cup B|} \geq t \geq 1/2$, a sufficiently good estimation of $\ell_p(x-y)^p$ leads to a good estimation of the Jaccard similarity. In principle, any ℓ_p sketch could then be used to estimate the above quantity. By employing the most efficient ℓ_2 sketch available [275], Bachrach and Porat obtained a $(1 \pm \varepsilon)$-approximation to the similarity of two item sets with d dimensional features with $O(\frac{(1-t)^2}{\varepsilon^2} \log d)$ bits of space and constant update time when the similarity is assumed to be at least $1/2$. This algorithm is readily implementable in streams, however it does not seem to support the fast candidate generation of min-hashing based approaches.

4.2 Our Contribution

Our main focus is to design a data structure that supports the filtering approach by Cohen et al [106] and operates in dynamic data streams. Dynamic streams are particularly relevant for association rule mining applications considered by Cohen et al. [106] and recommender systems considered by Bachrach and Porat [41], as the item sets may be adjusted and modified over time. We operate in a semi-streaming model, i.e. we aim to use only n polylog$(n|U|)$ bits of space, and the stream consists of triples (i, j, k), where i signifies the item set, j the item, and $k = \{-1, 1\}$ indicates whether the item gets added or deleted.

Similarity Approximation In a first step, we want to be able to maintain a form of approximation of the similarity of two item sets in a dynamic stream while using little space. Unfortunately, determining whether two item sets have non-zero similarity requires $\Omega(|U|)$ space, see Pagh et al. [259]. Bachrach and Porat [41] avoid this by only considering large similarities. Like Bachrach and Porat, we also use the identity $\ell_p(a-b)^p = |A \triangle B|$, however we base our algorithms around ℓ_0 sketches, rather than ℓ_2. Although there are no space improvements for the regime of large similarities (i.e. the regime we are interested in), ℓ_0 sketches allow us to $(1 \pm \varepsilon)$-approximate the distance function $1 - \frac{|A \cap B|}{|A \cup B|}$, which is something the other ℓ_p sketches do not seem able to achieve. Indeed, the work by Chierichetti and Kumar [96] already hints that approximating the distance is necessary to support a locality-sensitive hashing scheme (see also Theorem 2.2.7). This also gives us an additive ε-approximation to the Jaccard index.

Filtering and Nearest Neighbor Searching In a second step, we design a filtering mechanism based on ℓ_0 sketching techniques. The most common algorithmic tool is to randomly sample bits of a vector, which in our case is a random set of items. These bits themselves satisfy

certain forms of sensitivity. Specifically, we can show that roughly $\log|U|$ appropriately chosen bits have a lopsided sensitivity guarantee as per the Definition 2.2.4 of Indyk and Motwani [196]. For certain rational set similarities, including Hamming similarity and Roger-Tanimoto similarity, this is already sufficient to be plugged into the candidate generation framework by Cohen et al. [106].

The filtering mechanism for other rational set similarities such as Jaccard requires more work. Here, the sampled items are only sensitive if they have been chosen depending on the support of two candidate item sets A and B. We therefore independently retain samples for various possible cardinalities for each item set. When we search for item sets similar to some set A, we first filter out all sets with too large or too small support and run a LSH on the set of indexes we know to be sensitive.

For the analysis, we only require Chebyshev's inequality. This allows us to employ 2-wise independent hash functions with many appealing properties (see also Thorup [274]). They can be evaluated quickly, are easy to implement and require little additional storage.

4.3 Similarity and Distance Approximation in Dynamic Streams

We start off by showing that any rational set similarity with an LSH can be $(1 \pm \epsilon)$-approximated in dynamic streams. We first restate the bounds on sketching the ℓ_0 norm of a vector in turnstile streams.

Theorem 4.3.1 (Theorem 10 of Kane, Nelson and Woodruff [205])**.** *There is a turnstile streaming algorithm for $(1 \pm \varepsilon)$-approximating $\ell_0(x)$ of a d-dimensional vector x using space $O(\frac{1}{\varepsilon^2}\log|U|)$, with $2/3$ success probability, and with $O(1)$ update and reporting times.*

We note that the exact space bounds of the ℓ_0 sketch by Kane, Nelson and Woodruff depends on the magnitude of the entries of the vector. The stated space bound is sufficient for our purposes as we are processing binary entries. Using their algorithm as a black box, we get the following.

Theorem 4.3.2. *Given a constant $0 < \varepsilon \leq 0.5$, two item sets $A, B \subseteq U$, and some rational set similarity $S_{x,y,z,z'}$ with metric distance function $D_{x,y,z,z'}$, there exists a turnstile streaming algorithm that maintains a $(1 \pm \varepsilon)$ approximation to $D_{x,y,z,z'}(A,B)$ with constant probability. The algorithm uses $O(\frac{1}{\varepsilon^2}\log|U|)$ space and each update and query requires $O(1)$ time.*

Proof. Observe that $|A \triangle B| = \ell_0(a-b)$ and $|A \cup B| = \ell_0(a+b)$, where a and b are the characteristic vectors of A and B, respectively. Since $Den_{x,y,z,z'}(A,B) - Num_{x,y,z,z'}(A,B) = (z'-z)\cdot|A \triangle B|$ is always non-negative due to $z' > z$, we only have to prove that $Den_{x,y,z,z'}(A,B)$ is always a non-negative linear combination of terms that we can approximate via sketches. First, consider the case $x \geq y$. Reformulating $Den_{x,y,z,z'}(A,B)$, we have

$$Den_{x,y,z,z'}(A,B) = y\cdot|U| + (x-y)\cdot|A \cup B| + (z'-x)\cdot|A \triangle B|.$$

Then both numerator and denominator of $D_{x,y,z,z'}$ can be written as a non-negative linear combination of $|U|$, $|A \triangle B|$ and $|A \cup B|$. Given a $(1 \pm \varepsilon/5)$ of these terms, we have an upper bound of $\frac{1+\varepsilon/5}{1-\varepsilon/5} \leq (1+\varepsilon/5) \cdot (1+2\varepsilon/5) \leq (1+\varepsilon)$ and a lower bound of $\frac{1-\varepsilon/5}{1+\varepsilon/5} \geq (1-\varepsilon/5)^2 \geq (1-2\varepsilon/5)$ for any $\varepsilon \leq 0.5$.

Now consider the case $x < y$. Using a different reformulation

$$Den_{x,y,z,z'}(A,B) \quad = \quad Den_{y,x,z,z'}(\overline{A},\overline{B}) = (y-x) \cdot |\overline{A} \cup \overline{B}| + x \cdot |U| + (z'-y) \cdot |\overline{A} \triangle \overline{B}|,$$

we can write the denominator as a non-negative linear combination of $|\overline{A} \triangle \overline{B}|$, d and $|\overline{A} \cup \overline{B}|$. Dynamic updates can maintain an approximation of $|\overline{A} \triangle \overline{B}|$ and $|\overline{A} \cup \overline{B}|$, leading to upper and lower bounds on the approximation ratio analogous to those from case $x \geq y$.

By plugging in the ℓ_0 sketch of Kane, Nelson, and Woodruff (Theorem 4.3.1), the theorem follows. □

Using a similar approach, we can approximate the distance of root similarity functions admitting a locality hashing scheme. Define $S^{\alpha}_{x,y,z,z'} := 1 - (1 - S_{x,y,z,z'})^{\alpha}$. We first give a characterization analogous to Theorem 2.2.7.

Theorem 4.3.3 (Theorem 4.8 and 4.9 of [96])**.** *The root similarity $S^{\alpha}_{x,y,z,z'}$ is LSHable if and only if $z' \geq \frac{\alpha+1}{2} \max(x,y)$ and $z' \geq z$.*

The next theorem can now be proven using ideas similar to Theorem 4.3.2.

Theorem 4.3.4. *Given a constant $0 < \varepsilon \leq 0.5$, two item sets $A, B \subseteq U$, and some LSHable root similarity $S^{\alpha}_{x,y,z,z'}$, there exists a turnstile streaming algorithm that maintains a $(1 \pm \varepsilon)$ approximation to $D^{\alpha}_{x,y,z,z'}(A,B) := 1 - S^{\alpha}_{x,y,z,z'}(A,B)$ with constant probability. The algorithm uses $O(\frac{1}{\varepsilon^2} \log |U|)$ space and each update and query requires $O(1)$ time.*

Proof. We consider the case $x \geq y$, the case $y \geq x$ can be treated analogously. Again we will show that we can $(1 \pm \varepsilon)$-approximate the denominator of $S_{x,y,z,z'}$; the remaining arguments are identical to those of Theorem 4.3.2.

Consider the following reformulation of the denominator

$$Den_{x,y,z,z'}(A,B) = y \cdot |U| + (x - z') \cdot |A \cap B| + (z' - y) \cdot |A \cup B|.$$

We first note that we can obtain an estimate of $|A \cap B|$ in a dynamic data stream with additive approximation factor $\frac{\varepsilon}{2} \cdot |A \cup B|$ by computing $|A| + |B| - \widehat{|A \cup B|}$, where $\widehat{|A \cup B|}$ is a $(1 \pm \varepsilon/2)$-approximation of $|A \cup B|$.

Due to Theorem 4.3.3, we have $x - z' \leq 2 \cdot z' - z' \leq z'$ and either $z' - y \geq \frac{z'}{2}$ or $y \geq \frac{z'}{2}$. Hence $\frac{\varepsilon}{2} \cdot (x - z') \leq \frac{\varepsilon}{2} \cdot z' \leq \varepsilon \cdot \max(z', (z' - y))$. Since further $|U| \geq |A \cup B|$, we then obtain a $(1 \pm \varepsilon)$-approximation to the denominator. □

Remark 4.3.5. Theorems 4.3.2 and 4.3.4 are not a complete characterization of distance functions induced by similarities that can be $(1 \pm \varepsilon)$-approximated in turnstile streams. Consider, for instance, the Sørenson-Dice coefficient $S_{2,0,0,1} = \frac{2 \cdot |A \cap B|}{|A|+|B|}$ with $1 - S_{2,0,0,1} = \frac{|A \triangle B|}{|A|+|B|}$. Neither is $1 - S_{2,0,0,1}$ a metric, nor do we have $z' \geq \frac{\alpha+1}{2} x$ for any $\alpha > 0$. However, both numerator and denominator can be approximated using ℓ_0 sketches.

The probability of success can be further amplified to $1 - \delta$ in the standard way by taking the median estimate of $O(\log(1/\delta))$ independent repetitions of the algorithm. Union bounding over $\binom{n}{2}$ pairs, we then get the following corollary.

Corollary 4.3.6. *Let S be a rational set similarity with ground set U and metric distance function $1-S$. Let further $N \subset \mathcal{P}$ be a collection of n item sets. Given a dynamic data stream consisting of updates of the form $(i, j, v) \in [n] \times [|U|] \times \{-1, +1\}$ meaning that $x^{(i)}_j = x^{(i)}_i + v$ where $x^{(i)} \in \{0, 1\}^{|U|}$, there is a streaming algorithm that can compute with probability at least $1 - \delta$ for all pairs (i, i')*

- *a $(1 \pm \varepsilon)$ multiplicative approximation of $1 - S(x^i, x^{i'})$ and*
- *an ϵ-additive approximation of $S(x^i, x^{i'})$.*

The algorithm uses $O(n \log(n/\delta) \cdot \varepsilon^{-2} \cdot \log |U|)$ space and each update and query needs $O(\log(n/\delta))$ time.

4.4 Locality Sensitive Hashing in Dynamic Streams

We note that despite the characterization of LSHable rational set similarities of Theorem 2.2.7, Corollary 4.3.6 does not imply the existence of a locality sensitive hashing scheme or even an approximate locality sensitive hashing scheme on the sketched data matrix.

In the following, we will present a simple dynamic streaming algorithm that possesses such a guarantee, albeit with weaker approximation ratios. Specifically, we want to find pairs of item sets with similarity greater than a parameter r_1, while we do not want to report pairs with similarity less than r_2. Our sensitivity bounds are in terms of Definition 2.2.4 by Indyk and Motwani [196]. The precise statement is given via the following theorem.

Theorem 4.4.1. *Let $0 < \varepsilon, \delta, r_1, r_2 < 1$ be parameters. Given a dynamic data stream over n item sets from a universe U, there exists an algorithm that maintains a $(r_1, r_2, (1 - \varepsilon)r_1, 6r_2/(\delta(1 - \varepsilon/5\sqrt{2r_1}))$-sensitive LSH for Jaccard similarity with probability $1 - \delta$. The algorithm uses $O(n \frac{1}{\varepsilon^4 \delta^5 \cdot r_1^2} \log |U| \log n|U|)$ space.*

The remainder of this section is split into two parts. First, we give a probabilistic lemma from which our sensitivity parameters can be derived. Second, we describe how the sampling procedure can be implemented in a streaming setting.

4.4.1 Sensitivity Bounds

While a black box reduction from any ℓ_0 sketch seems unlikely, we note that most ℓ_0 algorithms are based on bit-sampling techniques similar to those found in min-hashing. Our own algorithm is similarly based on sampling a sufficient number of bits or item indexes from each item set. Given a suitably filtered set of candidates, these indexes are then sufficient to infer the similarity. Let $U_k \subseteq U$ be a random set of elements where each element is included with probability 2^{-k}. Further, for any item set A, let $A_k = A \cap U_k$. Note that in $S_{x,y,z,z}(A_k, B_k)$ the value of $|U|$ is replaced by $|U_k|$. At the heart of the algorithm now lies the following technical lemma.

Lemma 4.4.2. *Let $0 < \varepsilon, \delta, r < 1$ be constants and $S_{x,y,z,z'}$ a rational set similarity with metric distance function. Let A and B be two item sets. Then the following two statements hold.*

1. *If $S_{x,y,z,z'}(A,B) \geq r$ and $k \leq \log\left(\frac{\varepsilon^2 \cdot \delta \cdot r \cdot Den_{x,y,z,z'}(A,B)}{100 \cdot z'}\right)$ we have*

$$(1-\varepsilon)S_{x,y,z,z'}(A,B) \leq S(A_k,B_k) \leq (1+\varepsilon)S_{x,y,z,z'}(A,B)$$

   with probability at least $1-\delta$.

2. *With probability at least $1-\delta$, we have*

$$S_{x,y,z,z'}(A_k,B_k) \leq \frac{2}{\delta(1-(\varepsilon/5)\cdot\sqrt{2r})} \cdot S_{x,y,z,z'}(A,B).$$

Proof. Let $Den_k = Den(A_k,B_k)$, $Num_k = Num(A_k,B_k)$, and $X_i = 1$ iff $i \in U_k$. If $S_{x,y,z,z}(A,B) \geq r$ then $Num(A,B) \geq r \cdot Den(A,B)$. Thus, we have $\mathbb{E}[Num_k] = Num(A,B)/2^k \geq r \cdot Den(A,B)/2^k$ and $\mathbb{E}[Den_k] = Den(A,B)/2^k$. Moreover, we have the following variance bound

$$\begin{aligned}
\mathbf{Var}\left[Den_k\right] &= \mathbf{Var}\left[(x-y)\cdot|A_k\cap B_k| + y\cdot|U_k| + (z'-y)\cdot|A_k \triangle B_k|\right] \\
&= \mathbf{Var}\left[x\cdot|A_k\cap B_k| + y\cdot(|U_k| - |A_k\cup B_k|)| + z'\cdot|A_k\triangle B_k|\right] \\
&= x^2\sum_{i\in A\cap B}\mathbf{Var}\left[X_i\right] + y^2\sum_{i\in U\setminus(A\cup B)}\mathbf{Var}\left[X_i\right] + z'^2\sum_{i\in A\triangle B}\mathbf{Var}\left[X_i\right] \\
&\leq \left((x^2-y^2)|A\cap B| + y^2\cdot|U| + (z'^2-y^2)|A\triangle B|\right)/2^k \\
&= \left((x+y)\cdot(x-y)|A\cap B| + y\cdot y\cdot|U| + (z'+y)\cdot(z'-y)|A\triangle B|\right)/2^k \\
&\leq \max\{x+y, z'+y, y\}\cdot\left((x-y)|A\cap B| + y\cdot d + (z'-y)|A\triangle B|\right)/2^k \\
&\leq 2z'\cdot\mathbb{E}\left[Den_k\right]
\end{aligned}$$

and analogously

$$\mathbf{Var}\left[Num_k\right] \leq \max\{x+y, z+y, y\}\cdot\mathbb{E}\left[Num_k\right].$$

Using Chebyshev's inequality we have

$$\mathbb{P}\left[|Den_k - \mathbb{E}\left[Den_k\right]| \geq \varepsilon/5 \cdot \mathbb{E}\left[Den_k\right]\right] \leq \frac{50z'}{\varepsilon^2 \cdot \mathbb{E}\left[Den_k\right]} \leq \frac{50z' \cdot 2^k}{\varepsilon^2 \cdot Num(A,B)},$$

and

$$\mathbb{P}\left[|Num_k - \mathbb{E}\left[Num_k\right]| \geq \varepsilon/5 \cdot \mathbb{E}\left[Num_k\right]\right] \leq \frac{25\max\{x+y, z+y, y\}}{\varepsilon^2 \cdot \mathbb{E}\left[Num_k\right]} \leq \frac{50z' \cdot 2^k}{\varepsilon^2 \cdot Num(A,B)}.$$

If $k \leq \log\left(\frac{\varepsilon^2\delta Num(A,B)}{100z'}\right) \leq \log\left(\frac{\varepsilon^2\delta r Den(A,B)}{100z'}\right)$ then both $|Den_k - \mathbb{E}\left[Den_k\right]| \leq \frac{\varepsilon}{5}\mathbb{E}\left[Den_k\right]$ and $|Num_k - \mathbb{E}\left[Num_k\right]| \leq \frac{\varepsilon}{5}\mathbb{E}\left[Num_k\right]$ hold with probability at least $1 - \delta/2$. Then we can bound $S_{x,y,z,z'}(A_k, B_k) = Num_k/Den_k$ from above by

$$\frac{Num(A,B)/2^k + \varepsilon Num(A,B}{Den(A,B)/2^k - \varepsilon Den(A,B)/2^k)/2^k} = \frac{1+\varepsilon/5}{1-\varepsilon/5} \cdot S_{x,y,z,z'}(A,B) \leq (1+\varepsilon) \cdot S_{x,y,z,z'}(A,B).$$

Analogously, we can bound $S_{x,y,z,z'}(A_k, B_k)$ from below by $\frac{1-\varepsilon/5}{1+\varepsilon/5} \cdot S_{x,y,z,z'}(A,B) \geq (1-\varepsilon) \cdot S_{x,y,z,z'}(A,B)$ which concludes the proof of the first statement.

For the second statement, we note that the expectation of Num_k can be very small because we have no lower bound on the similarity. Hence, we cannot use Chebyshev's inequality for an upper bound on Num_k. But it is enough to bound the probability that Num_k is greater than or equal to $(2/\delta) \cdot \mathbb{E}\left[Num_k\right]$ by $\delta/2$ using Markov's inequality. With the same arguments as above, we have that the probability of $Den_k \leq (1-\varepsilon') \cdot \mathbb{E}\left[Den_k\right]$ is bounded by $\frac{\varepsilon^2 r\delta}{25 \cdot \varepsilon'^2}$ which is equal to $\delta/2$ if $\varepsilon' = \varepsilon/5 \cdot \sqrt{2r}$. Putting everything together we have that

$$S_{x,y,z,z'}(A_k, B_k) \leq \frac{2}{\delta(1 - (\varepsilon/5) \cdot \sqrt{2r})} \cdot S_{x,y,z,z'}(A,B)$$

with probability at least $1 - \delta$. □

Applying this lemma on a few better known similarities gives us the following corollary. More examples of rational set similarities can be found in Naish, Lee, and Ramamohanarao [255].

Corollary 4.4.3. *For the following similarities, the following values of k are sufficient to apply Lemma 4.4.2:*

Similarity	*Jaccard*	*Hamming*	*Anderberg*	*Rogers-Tanimoto*
Parameters	$S_{1,0,0,1}$	$S_{1,1,0,1}$	$S_{1,0,0,2}$	$S_{1,1,0,2}$
Sampling Rate	$\log\left(\frac{\varepsilon^2\delta r^2\lvert A\rvert}{100}\right)$	$\log\left(\frac{\varepsilon^2\delta r^2\lvert U\rvert}{100}\right)$	$\log\left(\frac{\varepsilon^2\delta r^2\lvert A\rvert}{200}\right)$	$\log\left(\frac{\varepsilon^2\delta r^2\lvert A\rvert}{200}\right)$

4.4.2 Streaming Implementation

When applying Lemma 4.4.2 to a dynamic streaming environment, we have to address a few problems. First, we may not know how to specify the number of items we are required to sample. For Hamming and Rogers-Tanimoto similarities, it is already possible to run a black box LSH algorithm (such as the one by Cohen et al. [106]) if the number of sampled items are chosen via Corollary 4.4.3. For Jaccard (and Anderberg), the sample sizes depend on the cardinality of A, which requires additional preprocessing steps.

Cardinality-Based Filtering As a first filter, we limit the candidate solutions based on their respective supports. For each item, we maintain the cardinality, which can be done exactly in a dynamic stream via counting. If the sizes of two item sets A and B differ by a factor of at least r_1, i. e., $|A| \geq r_1 \cdot |B|$, then the distance between these two sets has to be

$$1 - S(A, B) = \frac{|A \triangle B|}{|A \cup B|} \geq \frac{|A| - |B|}{|A|} \geq 1 - 1/r_1.$$

We then discard any item set with cardinality not in the range of $[r_1 \cdot |A|, \frac{1}{r_1}|A|]$. Like the algorithm by Cohen et al [106], we can do this by sorting the rows or hashing.

Small Space Item Sampling Since the cardinality of an item set may increase and decrease as the stream is processed, we have to maintain multiple samples U_k in parallel for various values of k. If a candidate k is larger than the threshold given by Corollary 4.4.3, we will sample only few items and still meet a small space requirement. If k is too small, $|U_k|$ might be too large to store. We circumvent this using a nested hashing approach we now describe in detail.

We first note that U_k does not have to be a fully independent randomly chosen set of items. Instead, we only require that the events X_i are pairwise independent. The only parts of the analysis of Lemma 4.4.2 that could be affected are the bounds on the variances, which continue to hold for pairwise independence. This allows us to emulate the sampling procedure using universal hashing. Assume that $|U|$ is a power of 2 and let $h : [|U|] \to [|U|]$ be a 2-wise independent universal hash function, i.e. $\mathbb{P}[h(a) = i] = \frac{1}{|U|}$, for all $i \in [|U|]$. We set $U_k = \{i \in [|U|] \mid \text{lsb}(h(i)) = k\}$, where $\text{lsb}(x)$ denotes the first non-zero index of x when x is written in binary and $\text{lsb}(0) = \log |U|$. Since the image of h is uniformly distributed on $[|U|]$, each bit of $h(i)$ is 1 with probability 1/2, and hence we have $\mathbb{P}[\text{lsb}(h(i)) = k] = 2^{-k}$. Moreover, for any two i, j the events that $\text{lsb}(h(i)) = k$ and $\text{lsb}(h(j)) = k$ are independent.

Following Theorem 2.1.2, h requires $\log |U|$ bits of space. To avoid storing the entire domain of h in the case of large $|U_k|$, we pick, for each $k \in [\log |U|]$, another 2-wise independent universal hash function $h_k : [|U|] \to [c^2]$, for some absolute constant c to be specified later. For some $i \in [|U|]$, we first check if $\text{lsb}(h(i)) = k$. If this is true, we apply $h_k(i)$. For

the jth item set, we maintain set $B^j_{k,\bullet}$ of buckets $B^j_{k,h_k(i)}$ for all $k \in \{0, \dots \log |U|\}$ and $h_k(i) \in \{0, \dots, c^2 - 1\}$. Each such bucket $B^j_{k,h_k(i)}$ contains the sum of the entries hashed to it. This allows us to maintain the contents of $B^j_{k,h_k(i)}$ under dynamic updates. For the interesting values of k, i.e. $k \in \Theta(\log |A|)$, the number of indexes sampled by h will not exceed some constant c. This means that the sampled indexes will be perfectly hashed by h_k, i.e. the sum contained in $B^j_{k,h_k(i)}$ consists of exactly one item index. If k is too small, i.e. we sampled too many indexes, h_k has the useful effect of compressing the used space, as c^2 sums of multiple item indexes requires at most $O(\log n|U|)$ bits of space, whereas t individual item indexes require $t \log |U|$ bits of space. We can then generate the fingerprint matrix, for instance, by performing a min-hash on the buckets $B^j_{k,\bullet}$ and storing the index of the first non zero bucket. For a pseudocode of this approach, see Algorithms 4. Algorithm 5 describes an example candidate generation as per Cohen et al. [106].

Algorithm 4 Filter-Preprocessing

Initialize $s_j = 0$ for all $j \in [n]$, $B^{(j)}_{k,l} = 0$ for all $j \in [n], k \in [0, \dots, \log |U|], l \in [c^2]$
Let $h : [|U|] \to [|U|]$ be a 2-universal hash function
Let $h_k : [|U|] \to [c^2]$ be independent 2-universal hash functions with $k = 0, \dots, \log d$
Let $U_k = \{i \in [n] \mid \text{lsb}(h(i)) = k\}$ with $k = 0, \dots, \log |U|$
On update (j, i, v):
    $k = \text{lsb}(h(i))$
    $B^{(j)}_{k,h_k(i)} = B^{(j)}_{k,h_k(i)} + (-1)^{1+v} i$

Algorithm 5 Filter candidates

Let $I = \{0, \log(1/r_1), 2\log(1/r_1), \dots, \log |U|\}$
Let H_i be an empty list for $i \in I$
for all $j \in [n]$ **do**
    $s = \ell_0(x^{(j)})$
    for all $k \in [\log(r_1^2 \cdot p \cdot s), \log(p \cdot s)] \cap I$ **do**
        Add $(j, MinHash(B^{(j)}_{k,\bullet}))$ to H_k
    end for
end for
return $\{(j, j') \mid \exists k : (j, h), (j', h') \in H_k \text{ and } h = h'\}$

Before proving Theorem 4.4.1, we first remark on the space required by Algorithms 4 and 5.

Observation 4.4.4. *Algorithms 4 and 5 require at most $O(n \cdot c^2 \log |U| \log(n|U|))$ bits of space.*

Proof. For each of the n item sets, we have $\log |U|$ collections $B^{(j)}_{k,\bullet}$ of c^2 buckets. Each bucket contains a sum of at most n indexes in the range of $\{0, \dots U - 1\}$. The space required for each hash function is at most $\log |U|$ due to Theorem 2.1.2. □

Proof of Theorem 4.4.1. Fix items sets J and J' and let j, j' be the corresponding indexes for the sets J and J', respectively. Set $p = \frac{\varepsilon^2 \cdot \delta}{600}$. If $S(J, J') \geq r_1$ then $r_1 \leq |J|/|J'| \leq 1/r_1$, that is, we enter lines 5-6 of Algorithm 5.

Let 2^k be the largest power of 2 such that $k \leq \log(p \cdot |J' \cap J|) \leq \log(p \cdot r_1 \cdot |J' \cup J|)$, i.e. $\frac{1}{2}(p \cdot r_1 \cdot |J' \cup J|) \leq 2^k \leq \log(p \cdot r_1 \cdot |J' \cup J|)$. Let U_k be a subset of indexes as determined by line 4 of Algorithm 4 and define $J_k := U_k \cap J$ and $J'_k := U_k \cap J'$. In expectation $\mathbb{E}[|J_k \cup J'_k|] = |J \cup J'|/2^k$. By Markov's inequality, we have $|J_k \cup J'_k| \leq \frac{3}{\delta} \cdot |J \cup J'|/2^k \leq \frac{3600}{\varepsilon^2 \delta^2 \cdot r_1}$ with probability at least $1 - \delta/3$. By setting the number of buckets in the order of $c^2 = \frac{3}{\delta}|J_k \cup J'_k|^2 \in O\left(\frac{1}{\varepsilon^4 \delta^5 \cdot r_1^2}\right)$, the elements of $J_k \cup J'_k$ will be perfectly hashed by h_k with probability at least $1 - \delta/3$ (line 3 of Algorithm 4). Since deleting indexes where both vector entries are zero does not change the similarity, the probability that the smallest index in the collection of buckets $B^{(j)}_{k,\bullet}$ is equal to the smallest index in the collection of buckets $B^{(j')}_{k,\bullet}$ is equal to the similarity of J_k and J'_k. Thus we have

$$\mathbb{P}[MinHash(B^{(j)}_{k,\bullet}) = MinHash(B^{(j')}_{k,\bullet})] = S(J_k, J'_k).$$

If $S(J, J') \geq r_1$ we have by our choice of p and due to the first part of Lemma 4.4.2, $S(J_k, J'_k) \geq (1 - \varepsilon) \cdot S(J, J')$ with probability $1 - \delta/3$. If $S(J, J') \leq r_2 < r_1$, we have due to the second part of Lemma 4.4.2 $S(J_k, J'_k) \leq \frac{6}{\delta(1 - (\varepsilon/5) \cdot \sqrt{2r_1})} \cdot S(J, J) \leq \frac{6r_2}{\delta(1-(\varepsilon/5)\cdot\sqrt{2r_1})}$ with probability $1 - \delta/3$. Conditioning on all events gives us a $(r_1, r_2, (1-\varepsilon)r_1, 6r_2/(\delta(1 - \varepsilon/5\sqrt{2r_1}))$-sensitive LSH with probability $1 - \delta$.

To conclude the proof, we plug our choice of c^2 into Observation 4.4.4. □

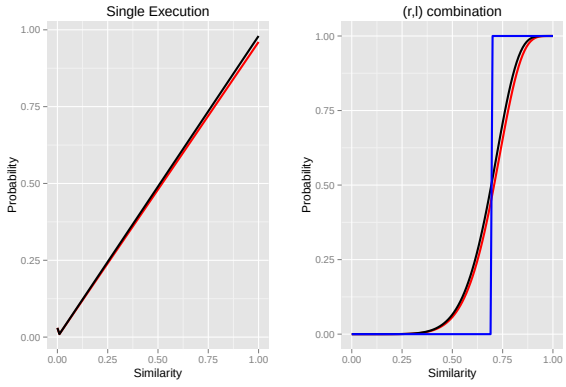

Figure 4.3: Selection probability (lower and upper bound) vs similarity for parameters $\delta = \varepsilon = r_1 = 0.01$, perfect hash probability of 0.99, $r = 7$, and $l = 10$. The threshold for the step function in the right figure is 0.7.

Note that if $S(A, B) > r_1$ then $\log(|A| \cdot p \cdot r_1) \leq \log(p \cdot |B|)$ and $\log(p/r_1 \cdot |A|) \leq \log(p \cdot |B|)$ which means there are hash values of both sets in some list H_k with $k \in I$ (in Algorithm 5) with good probability. The parameters in Theorem 4.4.1 can be chosen such that we are

able to use Algorithm 4 and Algorithm 5 similar to the min-hashing technique in the non-dynamic scenario. This also means that we can use similar tricks to amplify the probability of selecting high similar items in Algorithm 5 and lower the probability in case of a small similarity: Let $a, b \in \mathbb{N}$. Then we repeat the hashing part of Algorithm 5 a times and only add a pair to the output set iff all a hash values are equal. This procedure is repeated b times and the final output set contains all pairs which appear at least once in an output set of the b repetitions. The probability that a pair with similarity s is in the output set is $1-(1-p^a)^b$ with $p = (1-2\delta)(1 \pm \varepsilon)s$ if $s > c_2$ and $p \leq s/(\delta(1-\varepsilon/5\sqrt{c_2})$ otherwise. An example for some fixed parameters is given in Figure 4.3. Together with Theorem 4.3.2 we can approximately compute the distance (or similarity) of the pairs in the candidate output set of Algorithm 5.

4.5 Experimental Evaluation

Our primary goal was to measure the quality of the compression computed by Algorithms 4 and 5. We omitted a thorough evaluation of running times, as there exists many papers with experimental evaluations of min-hashing, see Henzinger [187], Cohen et al. [106, 107] and references therein. A particular focus was given on the performance when given little available space. In addition, we also aimed to find good combinations of bucket size (c^2) and sampling rates (α).

Setup We used the following setup for our experiments on both compression and running time. All computations were performed on two identical machines with the same hardware configuration (2.8 Ghz Intel E7400 with 3 MB L2 Cache and 8 GB main memory). The implementation was done in C++ and compiled with gcc 4.8.4 and optimization level 3. Each run was repeated 10 times. Our universal hash functions were generated as in Dietzfelbinger et al. [130] by drawing a non-negative odd integer a and a non-negative integer b. For a given key x, we computed the hash values via $a \cdot x + b$ modulo an appropriate domain. Otherwise the implementation follows that of Algorithms 4 and 5 with various choices of parameters. All random coin tosses were obtained from the random library [1].

Algorithms and Parametrization Our compression has two choices of parameters, namely the number of buckets c^2 in which we hash (hash function h_k in line 3 of Algorithm 4) and a parameter $\alpha \in (0, 1)$ specifying the relevant values of k in the algorithms. More precisely, for approximating the Jaccard similarity between two items A and B we chose $k = \log(\alpha \cdot (|A| + |B|)/2)$ and output $S(A_k, B_k)$. In Algorithm 5, we fixed $r_1 = 1/2$ and added a min-hash value to H_k with $k = s-1, s, s+1$ with $s = \lfloor \log(\alpha \cdot |A|) \rfloor$. The analysis indicates that the two parameters cannot be chosen independently of another, but it is nevertheless

[1] http://www.cplusplus.com/reference/random/

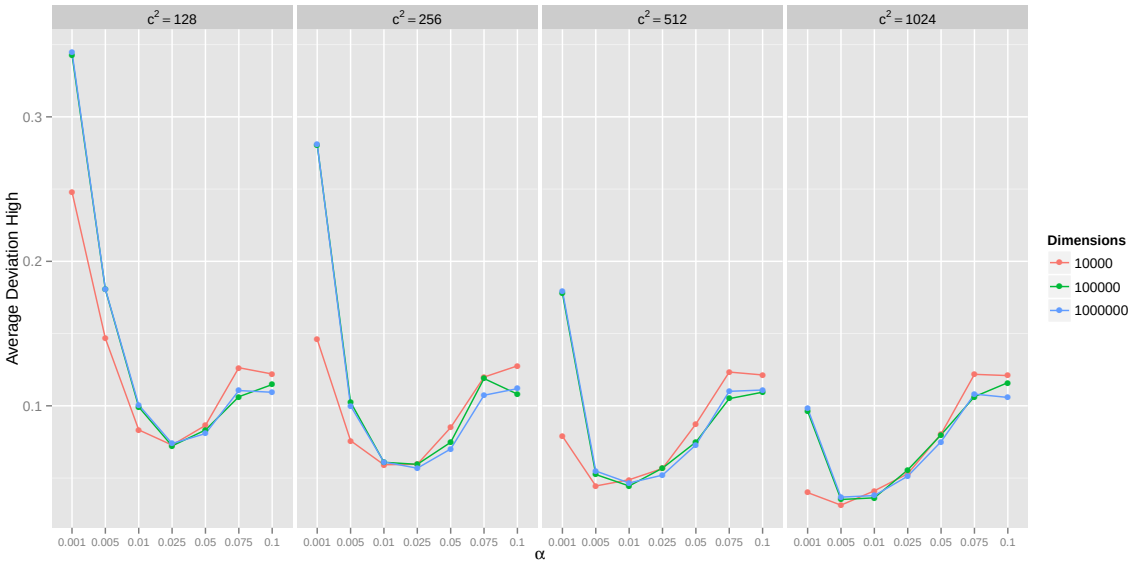

Figure 4.4: Mean deviation of high-similarity pairs ($S \geq 0.4$) for various parameters of Algorithm 4. Each instance was repeated 10 times.

important to know which range of parameters yield good results. For applications, it is likely that c is chosen to be as large as feasible, and α chosen to yield the best results for a given c. Our ranges for the number of buckets were $c^2 = \{128, 256, 512, 1024\}$ and for the sampling probability $\alpha = \{0.001, 0.005, 0.01, 0.05, 0.075, 0.1\}$.

Further, we implemented a locality sensitive hashing scheme for a faster evaluation of the Jaccard similarity, see also Cohen et al. [106]. We hashed the items and retained for each item set the smallest k indexes associated with items contained in the respective item set. In a second step, we partitioned the k indexes into b groups of a indexes. For each group, we produced a new key by concatenating the indexes and hashing the keys. With an appropriate choice of a and b, we have with good probability that two similar items with have at least one group with identical keys. For our sketches, we used the parameters that yielded the best compression results for the synthetic data sets.

Datasets An ideal evaluation of our algorithms would use data sets processed via a dynamic data stream, i. e., with insertions and deletion of items. While such streams frequently occur in practice, see for instance Mislove et al. [251], we only had access to final data sets. It could have been conceivable to create a synthetic dynamic stream along with dynamic data. We decided against this for multiple reasons. Firstly, we have little knowledge on the properties of dynamic stream in practice and therefore had no starting point to generate a benchmark. We are also not aware of any existing benchmark. Secondly, our algorithm is, to the best of our knowledge, the only algorithm applicable in this setting and therefore would have no

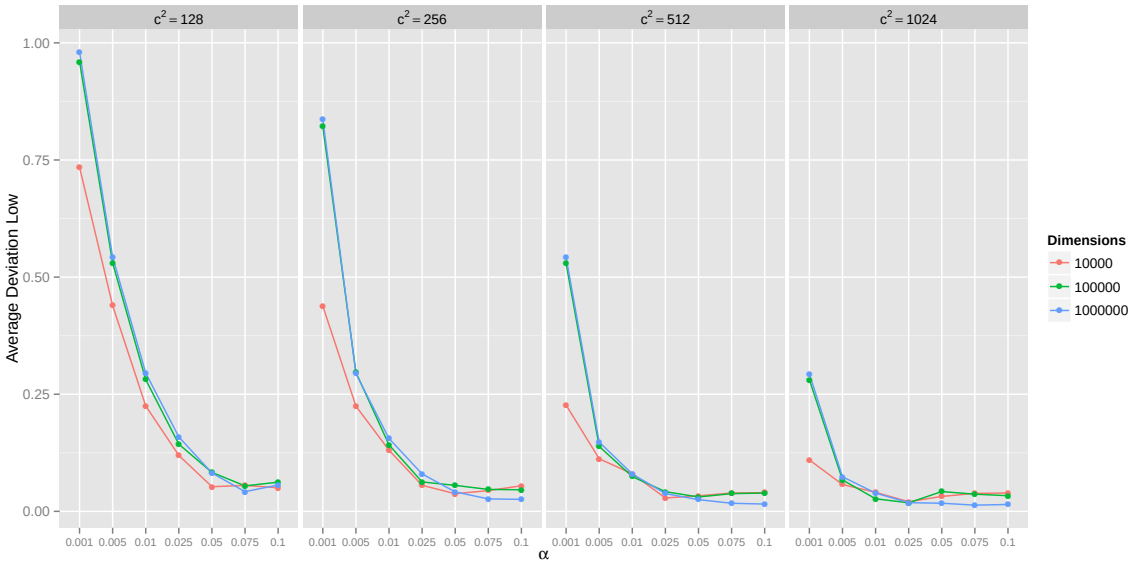

Figure 4.5: Mean deviation of low-similarity pairs for various parameters of Algorithm 4. Each instance was repeated 10 times.

other algorithm to compare to. Moreover, there is no technical difference between processing updates and deletions for our algorithm. If the final data set is identical, the only difference between an insertion only stream and a dynamic stream will be the respective length. As a result, our evaluation only considered final data.

We evaluated our approach both on synthetic data as well as on real-world data from an application using the Jaccard index as a similarity measure. For the synthetic data, we used the benchmark by Cohen et al. [106]. Here we are given a large binary data-matrix consisting of $10,000$ rows and either $10,000$, $100,000$ or $1,000,000$ columns. The rows corresponded to item sets and the columns to items, i. e., we compared the similarities of rows. Since large binary data sets encountered in practical applications are sparse, the number of non-zero entries of each row was between 1% to 5% chosen uniformly at random. Further, for every 100th row, we added an additional row of with higher Jaccard similarity in the range of $\{(0.35, 0.45), (0.45, 0.55), (0.55, 0.65), (0.65, 0.75), (0.75, 0.85), (0.85, 0.95)\}$. To obtain such a pair, we copied the preceding row (which was again uniformly chosen at random) and uniformly at random flipped an appropriate number of bits, e. g., for $10,000$ items, row sparsity of 5%, and similarity range (0.45, 0.55) we deleted an item contained in row i with probability $1/3$ and added a new item with probability $\frac{1}{19} \cdot \frac{1}{3} = \frac{1}{57}$.

The real-world data consists of features extracted from so-called PE files donated by G DATA [2]. The dataset is based on 2781 PE file samples from [265] where they were used for clustering to detect malware. G DATA extracted 714 categorical features which we converted

[2] https://www.gdatasoftware.com/

Similarity	0.0	0.05	0.1	0.15	0.2	0.25	0.3	0.35
Number of pairs	0	0	995	33864	364496	206572	233303	576286
Similarity	0.4	0.45	0.5	0.55	0.6	0.65	0.7	0.75
Number of pairs	861799	593181	549257	144769	33093	27777	42181	23185
Similarity	0.8	0.85	0.9	9.5				
Number of pairs	2617	7287	51042	113886				

Table 4.1: Distribution of similarity values from the G DATA data set.

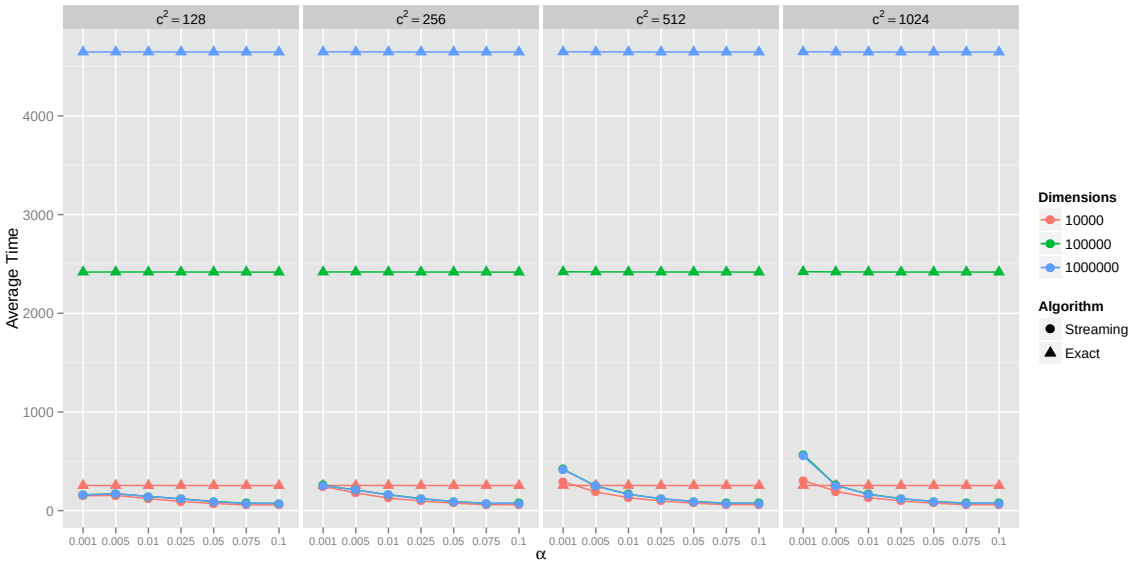

Figure 4.6: Runtime comparisons of the exact evaluation of the similarities on our summary and on original data set. The summary running times are the mean values of 10 repetitions.

to 18359 binary features. Each row in the final matrix has a support of 100-200 entries each. The distribution of similarities for this data set can be found in Table 4.1.

Results For compression we compared the exact similarities of the synthetic data set with the approximate similarities obtained for various parameterizations of our algorithm. We did not use min-hashing or any other filtering scheme to quickly evaluate the similarities on either original data set or our compression, as this introduces additional errors. As a result, the time required to evaluate the similarities is of secondary importance.

The major importance for this run of experiments was to find good combinations of c^2 and α and comparing the similarities. We measured the absolute deviation for high similarity (≥ 0.2) and low similarity (< 0.2) pairs separately. For a fixed bucket size, a larger value of α led to fewer items picked, while a smaller value of α led to many hash collisions. In

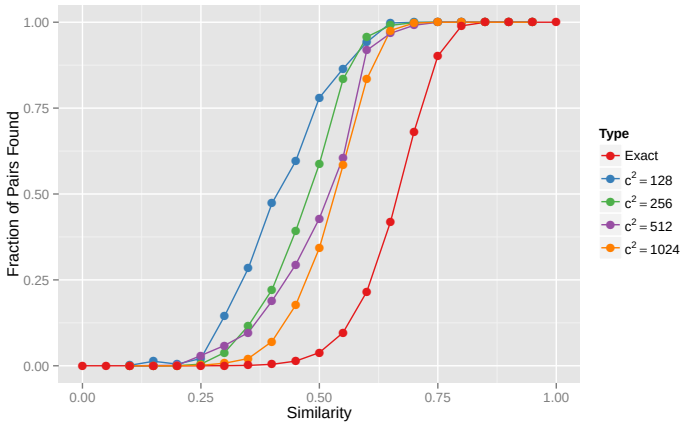

Figure 4.7: Thresholding for similarities at $r = 10$, $l = 40$ and $\alpha = 0.025$. With increasing bucket size, the theoretical curve marked in red was closer approximated by LSH computed on the sketch.

the former case, the compression performed worse on high-similarity pairs, while in the latter case the compression rates for both high and low-similarity pairs deteriorated, more so for the latter than the former, see also Figures 4.4 and 4.5. The best combinations of c^2 and α were different for high-similarity and low-similarity pairs. Good trade offs between both were achieved for 128 buckets with a sampling rate of $\alpha = 0.05$, 256 buckets with a sampling rate of $\alpha = 0.025$, 512 buckets with a sampling rate of $\alpha = 0.01$, and 1024 buckets with a sampling rate of $\alpha = 0.005$. On these average total deviation for these parameters was always below 0.1 and further decreased for larger bucket sizes. We note that these values of c^2 are below the theoretical bounds of Theorem 4.4.1, while having little to acceptable deviation for appropriately chosen values of α.

The time required to compute the sketch and thereafter evaluate the similarities was usually faster by a large magnitude (up to a factor of 10) compared to the original data set, see Figure 4.6. Generally, the fewer buckets and the lower the sampling rate, the faster the computation was carried out on the sketch. It should be noted that this already holds for relatively sparse data and since the sketch size is independent of the density, we would expect the improvement in time to be more apparent for denser data.

Lastly, we ran the LSH for the G DATA data set, again for various choices of parameters. Unlike for the synthetic data, there is no obvious correct threshold above which the relevant similarities lie. As a general rule, a large value of r moved the threshold towards 1, while a larger value of l moved the threshold to 0. By increasing both, the slope of the similarity curve increases. For a target threshold, i. e., fixed r and l and a fixed sampling rate α, the approximation to the theoretical similarity curve improved with an increasing number of buckets, see Figure 4.7.

5 Diameter and k-Center in Sliding Windows

In this chapter we present algorithms for diameter approximation and k-center clustering in sliding windows. The metric diameter problem is to find two points of maximum distance among a set of points lying in some metric space. Specifically, we obtain a $(3+\varepsilon)$ approximation algorithm for the diameter in arbitrary metric spaces while storing $O(\varepsilon^{-1} \log \alpha)$ points, where α is the aspect ratio. The same algorithm can be reused for the 2-center problem, in which case we obtain a 2-approximation. For both of these problems, we obtain matching conditional lower bounds for any sliding window algorithm with a non-polynomial dependency on the window size. For the k-center problem, we present a $(6+\varepsilon)$-approximation algorithm.

Our algorithm for the diameter (see Section 5.2) aims to find for each estimate of the value γ of the diameter two certificate points with distance greater than γ, while maintaining the two most recent points close to the two points forming the certificate. With every additional input point, we check whether we are able to update the certificate to a more recent timestamp.

We use a similar approach for the k-center problem. In the insert-only model, most previous results relied on doubling techniques. Given an estimate of OPT, a doubling approach draws balls of radius 2OPT around centers, opening a new center whenever a new point is not covered by the available centers. The main challenge is to provide lower bounds on OPT, and to preform an aggregation whenever an increase of OPT is detected. This becomes more difficult than in the insertion-only case, as OPT may now decrease with progression of time. Finding at least $k+1$ points at pairwise distance greater than $2 \cdot \gamma$ is done straightforwardly using a greedy doubling approach. For each point stored by the greedy doubling, we also maintain the newest point within radius 2γ. When a certificate of a lower bound expires, we then cluster all points kept in memory.

Our lower bounds assumes that any algorithm working in the metric oracle distance model is restricted to storing only input points. Given two points, we can query their distance via a call to the oracle. When processing the stream, an algorithm decides whether or not to store a point, or whether to discard it. Once discarded, a point cannot be recovered.

Under these assumptions, we are then able to insert an appropriately hard instance based on the points forgotten by the input. For randomized algorithms, we add additional points in which we hide a hard, randomly chosen instance for deterministic algorithms. A more in-depth description of our approach as well as a discussion on the generality of our results can be found in Section 5.4.

Notation According to Definition 2.5.1, the problem we study here is the metric k center problem where candidate centers are the input points, i.e. $A = F$. To query the distance between two points p and q, we invoke a distance oracle $\text{dist}(p, q)$. We assume that the oracle can accessed only for those points we currently keep in memory and that the oracle itself requires no additional space. The diameter is the maximum distance between any two points included in the current window and an α-approximate diameter are two points p, q included in the current window such that $\alpha \cdot \text{dist}(p, q)$ is greater or equal to the diameter. We note that in this model, expired points cannot be used by the algorithm or the optimal solution as centers or witnesses for the diameter, though they may be used in some other capacity.

5.1 Related Work

We covered related work for k-center in general and k-center in streaming models in Chapter 2. For completeness, we also review the state of the art for diameter approximation in streams. Feigenbaum et al. [143] were the first to consider the diameter in the sliding window model. For d dimensional Euclidean space, their algorithm uses $O\left(\left(\frac{1}{\varepsilon}\right)^{(d+1)/2} \log^3 N(\log\alpha + \log\log N + \frac{1}{\varepsilon})\right)$ bits of space. They also give a lower bound of $\Omega(\frac{1}{\varepsilon}\log N \log\alpha)$ for a $(1+\varepsilon)$ approximation factor in one dimension and, implicitly, an $\Omega(\log\alpha)$ space bound for any multiplicative approximation factor. This lower bound was later matched by Chan and Sadjad [89], who also gave an improved space bound of $O(\left(\frac{1}{\varepsilon}\right)^{(d+1)/2} \log\frac{\alpha}{\varepsilon})$ points for higher dimensions. For more general metric spaces, they obtain a $(2^{m+2} - 2 + \varepsilon)$ approximation with $O(N^{1/(m+1)})$ points.

In the metric distance oracle model there exists a folklore 2 approximation that maintains the first point p and the point with maximum distance from p. Guha [170] showed this algorithm to be essentially optimal, as no algorithm storing less than $\Omega(N)$ points can achieve a ratio better than $2-\varepsilon$ for any $\varepsilon > 0$. For Euclidean spaces, the best streaming algorithm with a polynomial dependency on d is due to Agarwal and Sharathkumar [5] with an almost tight approximation ratio of $\sqrt{2}+\varepsilon$ in $O(d\varepsilon^{-3}\log(1/\varepsilon))$ space. Agarwal et al. [4] proposed a $(1+\varepsilon)$-approximation using $O(\varepsilon^{-(d-1)/2})$ points. Similar space bounds have also been proposed for dynamic streams, see Andoni and Nguyen [20] and Chan [87]. For large d, Indyk [194] gave a sketching scheme with approximation factor $c > \sqrt{2}$ and space $O(dN^{1/(c^2-1)}\log n)$.

5.2 The Metric Diameter Problem

For a given estimate γ of the diameter, our algorithm for the metric diameter problem either produces two witness points at distance greater than γ or a point c that has a certain degree of centrality among the points in the current window. More formally, all points of the window inserted up to the insertion time of c will be proven to have distance at most 2γ from one

Algorithm 6 Sliding Window Algorithm for $(\gamma, 3 \cdot \gamma)$-gap Diameter

```
1:  c_old, q, r ← first point of the stream;
2:  c_new ← null;
3:  for all element p of the stream do
4:      if certificate point c_old expires then
5:          if (c_new ≠ null ∧ c_old = q) then
6:              c_old ← r; c_new ← null;
7:          if (c_new ≠ null ∧ c_old ≠ q) then
8:              c_old ← q; c_new ← null;
9:          if c_new = null then
10:             c_old ← r;
11:     INSERT(p);
12:     r ← p;
13: procedure INSERT(p)
14:     if c_new = null then
15:         if dist(p, r) > γ then
16:             c_old, q ← r; c_new ← p;
17:         else if dist(p, c_old) > γ then
18:             q ← r; c_new ← p;
19:     else
20:         if dist(p, r) > γ then
21:             c_old, q ← r; c_new ← p;
22:         else if dist(p, c_new) > γ then
23:             c_old ← c_new; q ← r; c_new ← p;
24:         else if dist(p, q) > γ then
25:             if c_old ≠ q then
26:                 c_old ← q; q ← r; c_new ← p;
```

another and points inserted after c will have distance at most γ from c. Thus, the diameter is at most 3γ.

Specifically, Algorithm 6 aims at maintaining a certificate for the diameter consisting of two points c_{old} and c_{new} such that $\text{dist}(c_{old}, c_{new}) > \gamma$ and $TTL(c_{old}) < TTL(c_{new})$. In addition, we also store the point q submitted immediately prior to c_{new} and the most recent point r. When a new point arrives, we test whether, based on the points we currently keep in memory, we can produce two points each with a larger TTL than $TTL(c_{old})$ with distance more than γ. If we find such a pair, we update the points accordingly, if not we update r and possibly q.

The algorithm has two different states depending on whether it found a pair of points of distance more than γ or not. The first state is indicated by $c_{new} = \textbf{null}$ and corresponds to the case that no such pair of points has been found. In this case, the algorithm maintains the following invariant, which certifies that the diameter of the points in the sliding window is at most $3 \cdot \gamma$.

We first observe that c_{old} is always inside the sliding window.

Invariant 5.2.1. If $c_{new} = \textbf{null}$, the following statements hold:

a) For any points a, b with $0 \leq TTL(a), TTL(b) \leq TTL(c_{old})$, we have $\text{dist}(a, b) \leq 2 \cdot \gamma$.

b) For any point a with $TTL(a) > TTL(c_{old})$, we have $\text{dist}(a, c_{old}) \leq \gamma$.

The second state corresponds to the case that we discover two points c_{old} and c_{new} with distance more than γ and is indicated by $c_{new} \neq \textbf{null}$. Besides the obvious invariant that $TTL(c_{new}) > TTL(c_{old})$, we also have to maintain the following technical invariants that are required for a new assignment of c_{old} when it expires from the window.

Invariant 5.2.2. If $c_{new} \neq$ **null** then the following statements hold:

a) $\text{dist}(c_{old}, c_{new}) > \gamma$.

b) For any point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, we have $\text{dist}(a, c_{old}) \leq \gamma$.

c) For any point a with $TTL(c_{new}) < TTL(a)$, we have $\text{dist}(a, c_{new}) \leq \gamma$.

d) If $c_{old} \neq q$ then for any point a with $TTL(q) < TTL(a)$, we have $\text{dist}(a, q) \leq \gamma$.

We observe that all the invariants hold initially, i.e. before line 3 of the algorithm is executed the first time. We also observe that Invariants 5.2.2a)-d) only apply to points that appear after c_{old}. It suffices to focus on these points because we only change c_{old} to points that arrive later and so we maintain our certificate at least until the time when c_{old} expires (and so all earlier points are gone).

Lemma 5.2.3. *If $c_{new} =$ **null** and Invariant 5.2.1 is satisfied before* Insert *then the following statements hold:*

*1) If $c_{new} =$ **null** after line 12 then Invariant 5.2.1 is satisfied.*

*2) If $c_{new} \neq$ **null** after line 12 then Invariant 5.2.2 is satisfied.*

Proof. We never execute lines 19-26 of Insert. If $\text{dist}(p, r) > \gamma$ then $c_{new} \neq$ **null** and Invariant 5.2.2.a) holds due to line 16, Invariant 5.2.2.b) holds due to the fact that there exists no point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, Invariant 5.2.2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 5.2.2.d) holds due to $c_{old} = q$. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{old}) > \gamma$ then $c_{new} \neq$ **null**, and Invariant 5.2.2.a) holds due to line 18, Invariant 5.2.2.b) holds due to Invariant 5.2.1.b), Invariant 5.2.2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 5.2.2.d) holds due to $r = q$ (before line 12) and line 15. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{old}) \leq \gamma$ then Invariant 5.2.1 continues to be satisfied for all points with TTL smaller than p and for the point p due to line 17. □

Lemma 5.2.4. *If $c_{new} \neq$ **null** and Invariant 5.2.2 is satisfied before* Insert*, then Invariant 5.2.2 is satisfied after line 12.*

Proof. If $\text{dist}(p, r) > \gamma$ then Invariant 5.2.2.a) holds due to line 21, Invariant 5.2.2.b) holds due to the fact that there exists no point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, Invariant 5.2.2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 5.2.2.d) holds due to $c_{old} = q$. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{new}) > \gamma$ then Invariant 5.2.2.a) holds due to line 23, Invariant 5.2.2.b) and 5.2.2.d) hold due to Invariant 5.2.2.c) before Insert, and Invariant 5.2.2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{new}) \leq \gamma$, and $\text{dist}(p, q) > \gamma$

and $q \neq c_{old}$ then Invariant 5.2.2.a) holds due to line 26, Invariant 5.2.2.b) holds due to Invariant 5.2.2.d) before INSERT, Invariant 5.2.2.c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 5.2.2.d) holds due to $q = r$ and line 20. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{new}) \leq \gamma$, and $\text{dist}(p, q) > \gamma$ and $q = c_{old}$ or $\text{dist}(p, q) \leq \gamma$, the Invariants 5.2.2.a) - d) hold for all points except for the newest, for which the invariants hold due to lines 20, 22, 24 and 25. □

Lemma 5.2.5. *If $c_{new} =$ **null** and Invariant 5.2.1 is satisfied before the line 4, then it is satisfied before INSERT (line 11).*

Proof. If c_{old} does not expire, then the claim obviously holds. Otherwise we execute line 10. Let c'_{old} be the expired point. Then we have for any two points a, b with $TTL(c'_{old}) < TTL(a), TTL(b) \leq TTL(r)$ $\text{dist}(a, c'_{old}), \text{dist}(b, c'_{old}) \leq \gamma$ due to Invariant 5.2.1.b) before line 4 and hence $\text{dist}(a, b) \leq 2\gamma$. Invariant 5.2.1.b) follows from $c_{old} = r$. □

Lemma 5.2.6. *If $c_{new} \neq$ **null** and Invariant 5.2.2 is satisfied before the line 4, then one of the following statements holds before INSERT (line 11):*

*1) $c_{new} =$ **null** and Invariant 5.2.1 is satisfied.*

*2) $c_{new} \neq$ **null** and Invariant 5.2.2 is satisfied.*

Proof. If c_{old} does not expire the second claim immediately holds. Otherwise, we denote by c'_{old} the expired point. If $c'_{old} = q$, then we execute line 6. We set $c_{old} = r$, naturally satisfying Invariant 5.2.1.b). Further, at this time we have $TTL(c_{new}) = 1$, and for any two points a, b with $TTL(c_{new}) \leq TTL(a), TTL(b) \leq TTL(r)$ we have $\text{dist}(a, c_{new}), \text{dist}(b, c_{new}) < \gamma$ due to Invariant 5.2.2.c) before line 4 and hence $\text{dist}(a, b) \leq 2\gamma$, satisfying Invariant 5.2.1.a).

If $c'_{old} \neq q$, then we execute line 8. We set $c_{old} = q$. For any two points a, b with $TTL(c'_{old}) < TTL(a), TTL(b) < TTL(c_{new})$ $\text{dist}(a, c'_{old}), \text{dist}(b, c'_{old}) \leq \gamma$ due to Invariant 5.2.2.b) before line 4 and hence $\text{dist}(a, b) \leq 2\gamma$, satisfying Invariant 5.2.1.a). For all points a with $TTL(a) > q$, Invariant 5.2.1.b) after line 12 follows from Invariant 5.2.2.d) before line 4. □

The proof of the theorem is a direct consequence of the invariants but included for completeness.

Theorem 5.2.7. *Given a set of points A with aspect ratio α and a window of size N, there exists an algorithm computing a $3(1+\epsilon)$-approximate solution for the metric diameter problem storing at most $8/\epsilon \cdot \ln \alpha$ points. The update time per point is $O(\varepsilon^{-1} \log \alpha)$.*

Proof. For any given estimate γ, we either have two points at distance at least γ, or Invariant 5.2.1 holds. In the latter case, we can bound the maximum diameter of two points p and q via the following case analysis.

$TTL(p), TTL(q) \leq TTL(c_{old})$: Then $\text{dist}(p, q) \leq 2\gamma$.

$TTL(p) \leq TTL(c_{old}) < TTL(q)$**:** Then $\text{dist}(p,q) \leq \text{dist}(p,c_{old}) + \text{dist}(c_{old},q) \leq 3\gamma$.

$TTL(c_{old}) < TTL(p), TTL(q)$**:** Then $\text{dist}(p,q) \leq \text{dist}(p,c_{old}) + \text{dist}(c_{old},q) \leq 2\gamma$.

Now define an exponential sequence to the base of $(1+\varepsilon)$, such that any value between $\min \text{dist}(p,q)$ and $\max \text{dist}(p,q)$ is $(1+\varepsilon)$ approximated. For each power of $(1+\varepsilon)$, we run Algorithm 6. Let γ be the largest value for which one of the instances of Algorithm 6 returns two points. The next larger estimate $\gamma \cdot (1+\varepsilon)$ guarantees us no diameter of size $3(1+\varepsilon)\gamma$, proving an approximation guarantee of at most $\frac{3(1+\varepsilon)\gamma}{\gamma} = 3(1+\varepsilon)$ The memory usage of the algorithm consists of 4 points per instance of Algorithm 6 and $\log_{1+\varepsilon} \alpha = \frac{\ln \alpha}{\ln(1+\varepsilon)} \leq \frac{2}{\varepsilon} \ln \alpha$ instances. □

Remark 5.2.8. To adapt this algorithm for windows where the maximum number of points are time dependent (e. g., the diameter of all points seen in the last hour) rather than the last N points, we can simply decouple the insertion procedure from the deletion routine. Whenever a point we currently keep in memory expires, we execute lines (4-10) and whenever a new point arrives, we call the INSERT procedure and line 12. Neither the invariants nor the proofs are affected in any way by this change.

5.3 The k-Center Problem

A 4-Approximation for Metric 2 Center

We run Algorithm 6 and show that, in the case of $k = 2$, it outputs a solution of cost at most 4 times the optimal solution. More precisely, let γ be the smallest estimate such that Algorithm 6 produces one point c with $\text{dist}(q,c) \leq 2\gamma$ for any point q in the current window. Further let a and b be the two points at distance greater than $\frac{\gamma}{1+\varepsilon}$ output Algorithm 6 for the next smaller estimate. W.l.o.g let $\text{dist}(a,c) \geq \text{dist}(b,c)$. Then $\{a,c\}$ form a 4 approximation.

Theorem 5.3.1. *Given a set of points A with aspect ratio α and a window of size S, there exists an algorithm computing a $4(1+\epsilon)$-approximate solution for the 2-center problem storing at most $8/\epsilon \cdot \ln \alpha$ points. The update time per point is $O(\varepsilon^{-1} \log \alpha)$.*

Proof. For c, the conditions of Invariant 5.2.1 apply, i.e. for any point p in our current window we have $\text{dist}(p,c) \leq 2\gamma$. We now distinguish between two cases.

OPT $\geq \frac{\gamma}{2(1+\varepsilon)}$**:** We have $\text{dist}(p,\{a,c\}) \leq \text{dist}(p,c) \leq 2\gamma \leq 4 \cdot (1+\varepsilon) \cdot \text{OPT}$.

OPT $< \frac{\gamma}{2(1+\varepsilon)}$**:** We first observe that a and b each fall into distinct clusters as their pairwise minimum distance is at least $\frac{\gamma}{1+\varepsilon}$. If a and c lie in distinct clusters, we have a 2-approximate solution, so we assume this not be the case. Then $\text{dist}(a,c) \leq 2 \cdot \text{OPT}$ and by construction, $\text{dist}(a,c) \geq \text{dist}(b,c)$. Then for any point p in the same cluster as b we have $\text{dist}(p,b) \leq 2 \cdot \text{OPT}$ and hence $\text{dist}(p,c) \leq \text{dist}(p,b) + \text{dist}(b,c) \leq 2 \cdot \text{OPT} + 2 \cdot \text{OPT} = 4 \cdot \text{OPT}$.

The proof of the space bound is analogous to that of Theorem 5.2.7. □

6-Approximation for Metric k-Center

A high level description of our algorithm is as follows, see also Algorithm 7 for pseudocode. We maintain a set A of at most $k+1$ *attraction points*. For each attraction point a, we maintain the newest point $R(a)$ within radius 2γ as a *representative*, i.e. $R(a) = \operatorname{argmax}_{p:\ \mathrm{dist}(p,a)\leq 2\gamma} TTL(R(a))$. When an attraction point expires, the representative point remains in memory. Call the set of representative points whose attraction points expired, the *orphaned representatives* O, and the set of representative points whose attraction points are still in the current window *active representatives* R. A new point p may become an attraction point if its distance is greater than 2γ to any point in A upon insertion. If the cardinality of A is greater than k, we retain the newest $k+1$ attraction points of A and all points with a greater TTL than the minimum TTL of A.

When asked to provide a clustering, we iterate through all estimates and either provide a counter example, or find a clustering which is then guaranteed to be a $6(1+\varepsilon)$-approximation. Our set of centers C first consist of an arbitrarily chosen point $p \in A \cup R \cup O$. Thereafter we greedily add any point point $q \in A \cup R \cup O$ with distance $\mathrm{dist}(q, C) > 2\gamma$. If upon termination $|C| > k$, we have a certificate for OPT $> \gamma$ and move to the next higher estimate. The smallest estimate with $|C| \leq k$ is then guaranteed to be a 6 approximation.

We start by giving the space bound.

Algorithm 7 Sliding Window Algorithm for $(\gamma, 6 \cdot \gamma)$-gap k-Center

```
1:  A, R, O ← ∅;
2:  for all element p of the stream do
3:      if q ∈ O expires then
4:          O ← O \ {q};
5:      if a ∈ A expires then
6:          DeleteAttraction(a);
7:      Insert(p);
8:  procedure DeleteAttraction(a)
9:      O ← O ∪ {R(a)};
10:     R ← R \ {R(a)};
11:     A ← A \ {a};
12: procedure Insert(p)
13:     D ← {a ∈ A | dist(p, a) ≤ 2 · γ};
14:     if D = ∅ then
15:         A ← A ∪ {p}
16:         R(p) ← p
17:         R ← R ∪ {R(p)}
18:         if |A| > k + 1 then
19:             a_old ← argmin_{a∈A} TTL(a);
20:             DeleteAttraction(a_old);
21:         if |A| > k then
22:             t ← min_{a∈A} TTL(a);
23:             for all q ∈ O do
24:                 if TTL(q) < t then
25:                     O ← O \ {q};
26:     else
27:         for all a ∈ D do
28:             Exchange R(a) with p in R;
```

Lemma 5.3.2. *At any given time, the number of points kept in memory is bounded by at most* $3(k+1)$.

Proof. We number all attraction points we keep in memory via the sequence in which they arrived, i.e. a_1 is the first attraction point, a_2 the second, etc. Call this sequence S. Note that in this sequence a_1 also expires before a_2.

At any given time, we maintain at most $k+1$ attraction points A and $k+1$ active representative points R due to lines 22-25 and the subroutine DeleteAttraction (lines 11-15). What remains to be shown is that the number of orphaned representative points O also never exceeds $k+1$.

First, we show that $TTL(a_{i+k+1}) > TTL(R(a_i)) \geq TTL(a_i)$. We distinguish between two cases. If a_i expires, then a_{i+k+1} gets inserted after a_i exits the window, hence $TTL(a_{i+k+1}) > N+1+TTL(a_i)$ and $TTL(R(a_i))+N \leq TTL(a_i)$. Otherwise, a_i gets deleted via lines 18-20 in the exact same time step in which a_{i+k+1} got inserted, in which case the claim also holds.

Now consider any point of time and let j be the maximum index of any attraction point in S that has expired. By the above reasoning, any representative spawned by $a_{j-(k+1)}$ is no longer in memory, and the space bounds holds. □

Lemma 5.3.3. *Let P be a set of points in a given window, $\gamma > 0$ an estimate of the clustering cost, $A \cup R \cup O$ the set of points we currently keep in memory with $|A| \leq k$. Then* $\max_{q \in P} \operatorname{dist}(p, R \cup O) \leq 4\gamma$.

Proof. We note that for any attraction point a, the representative $R(a)$ has maximum TTL among all points with distance at most 2γ. When a point p arrives, it has distance at most 2γ to some attraction point (which may be identical to p if we create a new one). Hence, if $R(a)$ is still in memory, the claim holds for p.

We now argue that by executing lines 18-25, all points p with $\operatorname{dist}(p, R \cup O) > 4\gamma$ have $TTL(p) < \min_{a \in A} TTL(a)$. If $TTL(p) > \min_{a \in A} TTL(a)$, then there exists an attraction point a' such that $\operatorname{dist}(p, a') \leq 2\gamma$. Then we have $TTL(R(a')) \geq TTL(p) > \min_{a \in A} TTL(a)$ and $\operatorname{dist}(p, R(a')) \leq 4\gamma$. Due to lines 24-25, $R(a')$ is never deleted. □

Combining these lemmas and using arguments analogous to those of the proof of Theorem 5.2.7, we have:

Theorem 5.3.4. *Given a set of points P with aspect ratio α and a window size N, there exists an algorithm computing a $6(1+\epsilon)$-approximate solution for the metric k-center problem storing $6(k+1)\ln(\alpha)/\epsilon)$ points. The update time per point is $O(k\varepsilon^{-1}\log\alpha)$.*

Proof. Again define an exponential sequence to the base $(1+\varepsilon)$ and run Algorithm 7 in parallel for all powers of $(1+\varepsilon)$ as objective value estimates. The space bound then follows from Lemma 5.3.2.

For each estimate γ, we greedily compute a clustering of $A \cup R \cup O$ where the pairwise distance between centers is greater than 2γ. Now consider the smallest estimate γ' for which the greedy clustering requires at most k centers C.

We have $\max_{p \in A \cup R \cup O} \text{dist}(p, C) \leq 2\gamma'$. We further have for any point q in the current window $\max_{q \in P} \text{dist}(q, C) \leq \max_{q \in P} \text{dist}(q, R \cup O) + \max_{p \in A \cup R \cup O} \text{dist}(p, C) \leq 4\gamma' + 2\gamma' \leq 6\gamma'$ due to Lemma 5.3.3. Since we have $\text{OPT} > \frac{\gamma'}{1+\varepsilon}$, C is a $6(1+\varepsilon)$ approximation. □

5.4 Lower Bounds

Our lower bounds for the studied problems hold for the metric oracle distance model. Whenever we wish to know the distance between two points p, q, we have to store the points in their entirety in order to invoke the oracle. The fundamental assumption used in the proofs of this section is that the algorithm cannot create new points, unlike, for instance, in Euclidean spaces, where we can store projections, means and similar points. In particular, this implies that once a point is discarded by the algorithm, it cannot be recalled by any means at a later date. Without any assumptions as to how the points are encoded, we measure the space complexity of an algorithm via the number of stored points, rather than the number of bits. We do not consider the space required to store the distance oracle, the TTL of each point or any other information we might wish to store. A similar reasoning can be also found in the paper by Guha [170], where the author was able to derive a lower bound of $\Omega(k^2)$ points for any deterministic single-pass streaming algorithm approximating the cost of the optimal k-center clustering up to a factor $2 + 1/k$.

Theorem 5.4.1. *For windows of size N, any deterministic sliding window algorithm outputting a solution of cost greater than $\frac{1}{3}OPT$ for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt{N})$ points.*

Proof. For the sake of contradiction, we assume that there exists an algorithm A that returns a solution whose cost is a factor 3 from the optimal solution while the algorithm stores less than $\sqrt{N}$ points. We start with the adversarial input sequence by submitting $\sqrt{N}$ *buckets* each containing $\sqrt{N}$ points. We denote the ith bucket by B_i and the jth point of bucket B_i by $p_{i,j}$ with $i, j \in \{0, \ldots, \sqrt{N} - 1\}$. Any point of bucket B_i is only read after all $\sqrt{N}$ points of bucket B_{i-1} are received. The points within each bucket are at distance 1 from one another.

Since the algorithm stores less than $\sqrt{N}$ points, there is at least one point in a bucket that must be discarded before a point of the next bucket is read. Let f_i be such a point for bucket B_i. The points of bucket B_i have the following distance to all future points:

$$\text{dist}(p_{i,j}, p_{i',*}) = \begin{cases} 2 & \text{for all } p_{i,j} = f_i \\ 1 & \text{otherwise} \end{cases}$$

for all $i' > i$.

It is easy to see that the distances satisfy the triangle inequality. Since there are $\sqrt{N}$ buckets and the algorithm stores less than $\sqrt{N}$ points, there is at least one bucket for which all the points are missing. Let B_t be this bucket.

We now introduce the $N+1$st input point p with distances

$$\operatorname{dist}(p, p_{i,j}) = \begin{cases} 1 & \text{if } i > t \\ 3 & \text{if } p_{i,j} = f_t \\ 2 & \text{otherwise} \end{cases}.$$

To show that the distances still satisfy the triangle inequality, we first observe that only $\operatorname{dist}(p, f_t)$ is neither 1 or 2 and thus requires special consideration. Here, we have $3 = \operatorname{dist}(p, f_t) \leq \operatorname{dist}(p, p_{i,j}) + \operatorname{dist}(p_{i,j}, f_t) = 1 + 2$ for $i > t$, and $3 = \operatorname{dist}(p, f_t) \leq \operatorname{dist}(p, p_{i,j}) + \operatorname{dist}(p_{i,j}, f_t) \leq 2 + 1$ for $i \leq t$.

Now, we keep inserting copies of p at distance 1 from one another until all the points of buckets $t' < t$ have expired. Since all the remaining points in memory are a subset of $\bigcup_{t'>t}(B_{t'} \setminus f_{t'})$ plus copies of p, the points in memory are all at distance 1 from one another. It follows that the algorithm can only output a solution of value 1 whereas the pair (p, f_t) induces a solution of value 3. □

We note that an algorithm by Chan and Sadjad [89] achieves a $2^{m+2} - 2 + \varepsilon$ approximation using $O(N^{1/(m+1)} \log \alpha)$ points, and it also falls under the same computational restrictions for the algorithms of this lower bound. Therefore, this lower bound cannot be strengthened by much, as their algorithm achieves a better approximation than 3 using roughly $N^{0.76}$ points by setting $m < \log 5/4$.

To utilize this instance for randomized algorithms, we require two modifications. First, we add additional points per bucket and uniformly choose f_i such that a randomized algorithm has little chance of retaining the correct point per bucket. Second, we use p (and its copies) to uniformly select a bucket B_t which the algorithm will have discarded with good probability.

Theorem 5.4.2. *For windows of size N, any randomized sliding window algorithm outputting a solution of cost greater than $\frac{1}{3}OPT$ with probability greater than $\frac{1}{2}$ for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt[3]{N})$ points.*

Proof. For ease of exposition, we will use a window of size $\Theta(N)$. The theorem then follows by rescaling N. We use $4N^{1/3}$ buckets consisting of $32N^{2/3}$ points each. In the following, any distance that is not further specified is set to be 1.

We iteratively replace one randomly chosen point from bucket B_i with f_i where f_i has distance 2 to any point from bucket B_j with $j > i$. At the end of the stream, we insert a point p with the following distances. First, choose a random bucket B_t and set $\operatorname{dist}(p, f_t) = 3$

and $\text{dist}(p,q) = 2$, where $q \in B_t \setminus \{f_t\}$. Any point inserted after bucket B_t has distance 1 to p and any point inserted before B_t has distance 2. We then repeatedly add copies of p at total of N times.

To show that the distances still satisfy the triangle inequality, we first observe that only $\text{dist}(p, f_t)$ is neither 1 or 2 and thus requires special consideration. Here, we have $3 = \text{dist}(p, f_t) \leq \text{dist}(p,q) + \text{dist}(q, f_t) = 1 + 2$ for $q \in B_i$ with $i > t$, and $3 = \text{dist}(p, f_t) \leq \text{dist}(p,q) + \text{dist}(q, f_t) \leq 2 + 1$ for $q \in B_i$ with $i \leq t$.

At any given time, the algorithm has to output a pair of points whose distance is within a factor 3 of the diameter of the current window. Observe that if the algorithm did not store any of the replaced points $\{f_0, \ldots, f_{4N^{1/3}-1}\}$ and not any point of bucket B_t then the algorithm is not able to produce two points at distance greater than 1. Hence, by Yao's minimax principle, it is sufficient to bound the number of points used by any deterministic algorithm against the above input distribution.

We first bound the probability that the algorithm stores some point f_i. Call this event A. If we assume that the algorithm did not store any of the points $\{f_1, \ldots, f_i\}$ it follows that the points in bucket B_{i+1} all have the same distance to the stored points. This implies that we can assume that the decision which points of bucket B_{i+1} will be kept is already fixed. The probability that f_i is one of these points is bounded by the hypergeometric distribution with population $32 \cdot N^{2/3}$, $N^{1/3}$ samples and 1 success in both population and sample: $\frac{\binom{32 \cdot N^{2/3}-1}{N^{1/3}-1} \cdot \binom{1}{1}}{\binom{32 \cdot N^{2/3}}{N^{1/3}}}$. Then the probability that no f_i is stored for any of the $4 \cdot N^{1/3}$ buckets can be lower bounded by

$$1 - \mathbb{P}[A] \geq \left(1 - \frac{\binom{32 \cdot N^{2/3}-1}{N^{1/3}-1} \cdot \binom{1}{1}}{\binom{32 \cdot N^{2/3}}{N^{1/3}}}\right)^{4 \cdot N^{1/3}} = \left(1 - \frac{N^{1/3}}{32 \cdot N^{2/3}}\right)^{4 \cdot N^{1/3}} \geq 1 - \frac{1}{8} = \frac{7}{8}.$$

Now we bound the probability that the algorithm retains any point from bucket t upon submission, which we call event B. Again, conditioned on the event that A does not hold ($\overline{A}$), the buckets from which the algorithm stores at least one point are fixed. The probability that B_t is among the stored buckets again follows a hypergeometric distribution with population $4 \cdot N^{1/3}$, $N^{1/3}$ samples and 1 success in both population and sample. Therefore $\mathbb{P}[B|\overline{A}] = \frac{\binom{4 \cdot N^{1/3}-1}{N^{1/3}-1} \cdot \binom{1}{1}}{\binom{4 \cdot N^{1/3}}{N^{1/3}}} = \frac{1}{4}$. Since one of the events A or B has to hold for the algorithm to output a solution with approximation factor greater than $\frac{1}{3}$, the probability that an algorithm storing less than $N^{1/3}$ points produces a solution with the desired approximation guarantee is at most $\mathbb{P}[A \cup B] \leq \mathbb{P}[A] + \mathbb{P}[B] = \mathbb{P}[A] + \mathbb{P}[B|A] \cdot \mathbb{P}[A] + \mathbb{P}[B|\overline{A}] \cdot \mathbb{P}[\overline{A}] \leq 2 \cdot \mathbb{P}[A] + \mathbb{P}[B|\overline{A}] \leq \frac{2}{8} + \frac{1}{4} = \frac{1}{2}$. □

We only briefly describe the k-center lower bound. The instance is also divided into sufficiently large buckets, from which the algorithm is forced to discard one point each. The main difference with the previous proof will be that the distances between all the points (except

for a randomly chosen missing point f_t) are 2 and the distance from f_t to the more recent buckets is 4.

Theorem 5.4.3. *For windows of size N, any randomized sliding window algorithm achieving an approximation factor less than 4 with probability greater than $\frac{1}{2}$ for the distance oracle metric 2 center problem with constant aspect ratio requires $\Omega(\sqrt[3]{N})$ points.*

Proof. Again, for ease of exposition, we will use a window of size $\Theta(N)$, the theorem then holds by rescaling N. We first pick a constant $\ell = 32$. Now, define $8N^{1/3}$ buckets consisting of $128N^{2/3} + \ell$ points each. The points of the ith bucket are at distance 2 from each other. We iteratively replace one randomly chosen point of the last $128N^{2/3}$ points from bucket B_i with f_i, where $\operatorname{dist}(p_{i,j}, p_{i',*}) = \begin{cases} 4 & \text{for all } p_{i,j} = f_i \\ 2 & \text{otherwise} \end{cases}$ for all points $i' > i$.
We now define ℓ points $p_0, \ldots, p_{\ell-1}$ whose distances are specified below but we do not insert them yet. Finally, we randomly choose a bucket t and a point $p* \in \{p_0, \ldots, p_{\ell-1}\}$ such that for all $p_{i'} \in \{p_0, \ldots, p_{\ell-1}\} \setminus \{p*\}$ and point p, we have

$$\operatorname{dist}(p_{i'}, p) = \begin{cases} 4 & \text{if } p = f_t \\ 1 & p \in \{p_0, \ldots, p_{\ell-1}\} \setminus \{p_{i'}\} \\ 2 & p \in \left(\bigcup_{t' \geq t} B_{t'}\right) \setminus \{f_t\} \end{cases} \text{ and } \operatorname{dist}(p*, p) = \begin{cases} 3 & \text{if } p = f_t \\ 1 & p \in \left(\bigcup_{t' \geq t} B_{t'}\right) \setminus \{f_t\} \end{cases}.$$

Let T_t denote the time at which the first point of B_t is inserted. Note that we do not specify the distance from the points of $\{p_0, \ldots, p_{\ell-1}\}$ to the buckets $B_{t''}$ for $t'' < t$, since we insert copies of the last point of the last bucket in order to make all the points of buckets $0, \ldots, t-1$ expire. Then we insert the points $p_0, \ldots, p_{\ell-1}$.

Several things should be noted about the input. First, all distances obey the triangle inequality. Second, f_t expires after time $T_t + \ell$. And lastly, at any given time from $T_t + \ell$ until f_t expires, there exists a solution of cost 1 which consists of $p*$ and f_t.

Moreover, remark that in order to obtain an approximation ratio better than 4, any algorithm has to be able to open centers at a point of B_t or at $p*$. By Yao's minimax principle, it is sufficient to bound the number of points used by any deterministic algorithm against the above input distribution.

We first bound the probability that the algorithm stores some point f_i. Call this event A. Assuming that the algorithm never stored a point $f_{i'}$, the points stored by any future bucket B_i with $i > i'$ are fixed. The probability that f_i is one of these points is bounded by the hypergeometric distribution with population $128N^{2/3}$, $N^{1/3}$ samples and 1 success in both population and sample: $\frac{\binom{128N^{2/3}-1}{N^{1/3}-1} \cdot \binom{1}{1}}{\binom{128N^{2/3}}{N^{1/3}}}$. Then the probability that no f_i is stored for any of

the $8N^{1/3}$ buckets can be lower bounded as follows.

$$\begin{aligned} 1-\mathbb{P}[A] &\geq \left(1-\frac{\binom{128N^{2/3}-1}{N^{1/3}-1}\cdot\binom{1}{1}}{\binom{128N^{2/3}}{N^{1/3}}}\right)^{8N^{1/3}} = \left(1-\frac{N^{1/3}}{128N^{2/3}}\right)^{8N^{1/3}} \geq 1-\frac{1}{16}=\frac{15}{16} \\ \Leftrightarrow \mathbb{P}[A] &\leq \frac{1}{16}. \end{aligned}$$

Now we bound the probability that the algorithm retains any point from bucket t upon submission, which we call event B. Again, conditioned on the fact that event A does not hold ($\overline{A}$), the buckets from which the algorithm stores at least one point are fixed. The probability that B_t is among the stored buckets again follows a hypergeometric distribution with population $8N^{1/3}$, $N^{1/3}$ samples and 1 success in both population and sample. Therefore $\mathbb{P}[B|\overline{A}] = \frac{\binom{8N^{1/3}-1}{N^{1/3}-1}\cdot\binom{1}{1}}{\binom{8N^{1/3}}{N^{1/3}}} = \frac{1}{8}$.

Finally, we consider the event that the algorithm does pick $p*$ as a center. Let C denote this event. In order for C to happen, the algorithm has to be able to distinguish between points of $p_0,\ldots,p_{\ell-1}$ or it has to pick $p*$ randomly from $p_0,\ldots,p_{\ell-1}$. The first case is identical to event B, the second case happens with probability at most $2/\ell$, since the algorithm can open 2 centers. It follows that $\mathbb{P}[C] \leq \mathbb{P}[B] + 2/\ell$.

If none of the events A, B, or C hold, the algorithm will not output two centers with approximation factor less than 4, The probability that an algorithm storing $N^{1/3}$ points achieves this is at most

$$\begin{aligned} \mathbb{P}[A\cup B\cup C] &\leq \mathbb{P}[A]+\mathbb{P}[B]+\mathbb{P}[C] \leq \mathbb{P}[A]+2\mathbb{P}[B]+2/\ell \\ &= \mathbb{P}[A]+2\left(\mathbb{P}[B|A]\cdot\mathbb{P}[A]+\mathbb{P}[B|\overline{A}]\cdot\mathbb{P}[\overline{A}]\right)+2/\ell \\ &\leq 3\cdot\mathbb{P}[A]+2\mathbb{P}[B|\overline{A}]+2/\ell \leq \frac{3}{16}+\frac{2}{8}+\frac{2}{32}=\frac{1}{2}. \end{aligned}$$

□

6 1-Center Clustering in the Jaccard Metric

In this chapter we present hardness and algorithmic results of the Jaccard center problem. Formally, we are given a collection of n item sets, where each item set is a subset of some ground set U. For any two sets $X, Y \subseteq U$, we measure their distance via the Jaccard metric, i.e. $\text{dist}(X,Y) = D(X,Y) = 1 - J(X,Y) = |X \triangle Y|/|X \cup Y|$. The task is to find a subset C of U, such that $\max_{X \in A} D(X,C)$ is minimized.

With the notation introduced in Chapter 2, we are studying the 1-center clustering problem where the clients A are the collection of item sets and the candidate center set is $F = \mathcal{P}(U)$, i.e. the set of all subsets of U.

We show that the problem is NP-hard to solve exactly, even when the input item sets have cardinality 2. Since the Jaccard distance is a metric, any input point is a trivial 2-approximate solution, and it is easy to see that this bound is tight. We propose two algorithms for the problem. The first is a $(1+\varepsilon)$ approximation algorithm with running time $n^{O(\varepsilon^{-6})}$. The second is an exact algorithm parameterized by $k = \max_{X \in N} |X \triangle C|$, i.e. the maximum Hamming norm of input points and Jaccard center C, with running time $2^{O(k^3)} \cdot n \cdot |U|^3$. As a consequence of our hardness result, we can show that under the exponential time hypothesis [192] no fixed parameter tractable algorithm with parameter k and running time $2^{o(k)}$ and no polynomial time approximation scheme with running time $2^{o(\sqrt{1/\varepsilon})}$ can exist.

Lastly, we also briefly remark on the continuous version of the problem. Here the input points are non-negative d-dimensional real vectors and $J(X,Y) = \sum_{i=1}^{d} \min(X_i, Y_i) / \sum_{i=1}^{d} \max(X_i, Y_i)$. While the Jaccard median problem remains NP-hard for the continuous setting [98, 270], the center problem becomes solvable in polynomial time.

6.1 Related Work

Most of theoretical computer science research focused on hashing algorithms for fast similarity search, which we discussed in Chapters 1 and 4. The arguably most famous clustering paper with a strong focus on the Jaccard metric is the ROCK algorithm by Guha, Rastogi and Shim [175]. In theoretical computer science, there exists some previous work on the the Jaccard median, i.e. finding a item set that minimizes the sum of Jaccard distances. Späth [270] gave a structural result for continuous Jaccard measures, showing that the coordinates of the optimal median are coordinates of the input point set. Watson [283] gave a gradient descent

algorithm, albeit without any bounds on running time. More recently, Chierichetti et al. [98] showed that the binary Jaccard median, i.e. $(1, 1)$ clustering with the Jaccard metric, problem is NP-hard but also admits a PTAS. The hardness extends to the continuous Jaccard median problem due to the structural result by Späth, and Chierichetti et al. also gave a PTAS in this setting.

Hardness for 1-center problems in certain finite metrics have been established, including permutation metrics such as Kendall tau and Cayley distances [39, 62, 263], the edit distance on strings [126, 256] and the Hamming metric on strings [153, 228]. The latter problem, also known as the *closest string problem*, is one of the most widely studied center problems in computer science with numerous results on fixed parameter algorithms [148, 166, 239] and approximation algorithms [19, 158, 225].

6.2 Approach and Techniques

Weak coreset are a useful tool for the minimum enclosing ball problem in d-dimensional Euclidean spaces [44, 101, 224, 284]. Given a set of points A, an weak ε-coreset is a subset S of A such that the expansion of the minimum enclosing ball of S by a factor of $(1+\varepsilon)$ contains all points of A. The fact that there exist weak coresets whose size only depends on $\text{poly}(\varepsilon^{-1})$ is useful in the context of approximation algorithms, see for instance [45]. In some sense weak coresets can be regarded as a dimension reduction, as it is only important to consider a subspace with dimension far smaller than d. It is important to note that the approximation guarantee only holds for the optimal Euclidean 1-center. If we wanted to approximate the maximum distance from any point p to its furthest point in A, S must have size exponential in the dimension d [3].

Our own algorithms are motivated by this feature, although our guarantees are weaker than those of weak coresets. We start by describing our approach for the PTAS, and then return to a Jaccard-center analogue of weak coresets. We first consider a natural linear system induced by the following equations

$$|X \triangle C| \leq \widehat{\text{OPT}} \cdot |X \cup C|.$$

where C is the candidate center X is an input set. If there exists some set C with maximum Jaccard distance OPT to any input point, there exists a binary vector such that the above equation holds for all all input sets. It turns out that the LP relaxation can be efficiently rounded for a large range of inputs, namely when $\text{OPT} \cdot |X| \in \Omega(\varepsilon^{-2} \log n)$ for any input set X.

A quasi-polynomial time approximation scheme with running time $|U|^{\varepsilon^{-2} \log n}$ now follows by choosing an the set X with minimal $\text{OPT} \cdot |X|$ and iterating over all subsets H of the ground set U of size $O(\log n/\varepsilon^2)$ and determining the best solution among all centers $X \triangle H$. To obtain a PTAS, we aim to reduce the number of coordinates by considering multiple input

sets simultaneously.

If weak coresets with size $\text{poly}(\varepsilon^{-1})$ existed for the Jaccard center problem, we would be done. Firstly,the LP cannot be efficiently rounded, the symmetric difference between any two input sets as well as the optimum set is bounded by $O(\varepsilon^{-4} \log n)$. Then we could iterate over all subsets of size $\text{poly}(\varepsilon^{-1})$ of U of which one is a weak coreset, and then enumerate all possible solutions in time $|U|^{\text{poly}(\varepsilon^{-1})}$. However, such guarantees seem unlikely. The Jaccard distance can be isometrically embedded into (high dimensional) squared Euclidean space, see Gower and Legendre [165], but the weak coreset results do not seem to be applicable to the constrained set of solutions corresponding to embedded item sets.

Instead, we observe that that if we simultaneously consider the items of multiple sets $X_1, \ldots, X_m$, all of which have small symmetric distances to the optimum, the number of candidate subsets cannot increase, but may be reduced. This is made more precise via the notion of core-cover. We call a collection of sets $S \subset N$ a core-cover, if an optimal center C is (mostly) contained in $\bigcap_{X \in S} X$. Specifically, we say that S is an *α-core-cover* if $C \cap \bigcup_{X \in S} X$ is an α-approximate solution. An *anchored core-cover* further restricts the possible solutions by always containing the items in the intersection of all sets of the core-cover, i.e. $\bigcap_{X \in S} X \cup (C \cap \bigcup_{X \in S} X)$ is an α-approximate solution. Crucially, we show that the size of an $(1+\varepsilon)$-anchored core-cover is depends only on ε^{-1}. This allows us to determine by brute force in time $n^{\text{poly}(\varepsilon^{-1})}$ an anchored core-cover and enumerate all possible solutions $\bigcap_{X \in S} X \cup (C \cap \bigcup_{X \in S} X)$.

Having obtained a PTAS running in time $n^{\text{poly}(\varepsilon^{-1})}$, we might consider whether there exists an efficient PTAS running in time $\text{poly}(n, |U|) \cdot \exp(\varepsilon^{-1})$. Indeed this was our original reason for studying the FPT-problem, as we know that under complexity assumptions for fixed parameter algorithms, a problem does not admit an efficient PTAS if it is not in FPT, see Cesati and Trevisan [83]. We can show that Jaccard center is in FPT for the parameter $k = \max_{X \in N} |X \triangle C|$, leaving the question of whether there exists an efficient PTAS open. Core-covers are well defined for $\alpha = 1$, i.e. when we are trying to solve the problem optimally. In this case, the size of core-cover depends only k. The main technical difficulties are to show that we can efficiently construct an anchored core-cover. As was the case for the PTAS, for a given preliminary anchored core-cover M, we can compute an induced optimum via complete enumeration. If the induced optimum has distance at most OPT to all sets $X \in N$, we are done. Otherwise, any set violating this bound can be added to M. The improvement rate of each added set matches the non-constructive bounds used to show the existence of core-covers, ensuring that the algorithm terminates quickly.

6.3 Preliminaries

Throughout this chapter denote by OPT the minimum value of $\min_{C \subseteq U} \max_{X \in N} D(X, C)$. We will always assume $\emptyset \notin N$, i.e. that the empty set is not part of the input, as otherwise $\emptyset$ is a trivial optimal solution with maximum distance 1 if there exists at least one further point in N, and maximum distance 0 if $N = \{\emptyset\}$. Lastly, we will use the following easily verified facts frequently throughout the paper.

Fact 6.3.1. *Let $X, Y \subseteq U$ be two item sets. Then the following bounds hold:*

- $|X \cap Y| = (1 - D(X, Y)) \cdot |X \cup Y|$
- $|X| \geq (1 - D(X, Y)) \cdot |Y|$
- $|X \setminus Y| \leq D(X, Y) \cdot |X|$

Proof. The equality is a reformulation of $1 - |X \cap Y|/|X \cup Y| = D(X, Y)$. For the two inequalities we observe

- $|X| \geq |X \cap Y| = (1 - D(X, Y)) \cdot |X \cup Y| \geq (1 - D(X, Y)) \cdot |Y|$ and
- $D(X, Y) = \frac{|X \triangle Y|}{|X \cup Y|} = \frac{|X \setminus Y| + |Y \setminus X|}{|X| + |Y \setminus X|} \geq \frac{|X \setminus Y|}{|X|}$.

□

6.4 Hardness of Binary Jaccard Center

We reduce the problem of finding the optimum Jaccard center from vertex cover defined as follows.

Definition 6.4.1. Given a graph $G(V, E)$, a vertex cover is a set $K \subset V$ such that $e \cap K \neq \emptyset$ for any $e \in E$. The minimum vertex cover is the vertex cover of smallest cardinality.

Theorem 6.4.2. *Computing the optimum Jaccard center is NP-hard even if every $X \in N$ has cardinality at most 2.*

Proof. It is well known that computing the minimum vertex cover is NP-hard [157]. It remains NP-hard, when the vertex cover is promised to have cardinality at most $\frac{|V|}{2} - 2$, as we can simply add an isolated star with one central node and $|V| + 5$ remaining nodes.

Let K be a minimum vertex cover of cardinality at most $\frac{|V|}{2} - 2$ in a graph $G(V, E)$ with no isolated nodes. Consider now the instance of the Jaccard center problem where the input item sets are E, the base set is V, and the center is some subset of V. We claim that a collection of vertexes C is an optimum Jaccard center if and only if C is a minimum vertex cover.

For every collection of vertices C and any edge $e \in E$, we have the following three cases:

$$D(e,C) = \begin{cases} 1 & \text{if } |C \cap e| = 0 \\ \frac{|C|}{|C|+1} & \text{if } |C \cap e| = 1 \\ \frac{|C|-2}{|C|} & \text{if } |C \cap e| = 2. \end{cases}$$

Note that the distance for some edge is 1 if and only if C is not a vertex cover. Note also that $\frac{|C|}{|C|+1} > \frac{|C|-2}{|C|}$, i.e. if $C \neq V$ then $\max_{e \in E} D(e,C) = \frac{|C|}{|C|+1}$. Now for any collection of vertices C that is a vertex cover with $|C| > |K|$, we have two cases. If $C \neq V$, then

$$\max_{e \in E} D(e,C) = \frac{|C|}{|C|+1} \geq \frac{|K|+1}{|K|+2} > \frac{|K|}{|K|+1} = \max_{e \in E} D(e,K).$$

If $C = V$, then

$$\max_{e \in E} D(e,V) = \frac{|V|-2}{|V|} = \frac{\frac{|V|}{2}-1}{\frac{|V|}{2}} \geq \frac{|K|+1}{|K|+2} > \frac{|K|}{|K|+1} = \max_{e \in E} D(e,K).$$

□

Corollary 6.4.3. *There exists no FPTAS for the binary Jaccard center problem unless P=NP.*

Proof. Two non-equal distances are at least apart by $\frac{1}{|U|^2}$. If an FPTAS were to exist, we could compute determine a $(1+\frac{1}{|U|^2})$ approximation in polynomial time. This approximation, however, would coincide with the optimal solution. □

Assuming the exponential time hypothesis (ETH), we can give stronger time bounds for PTAS and FPT. ETH, formulated by Impagliazzio, Paturi, and Zane [192] assumes that there exists some positive real number s such that 3-SAT with n variables and m clauses cannot be decided in time $2^{s \cdot n}(n+m)^{O(1)}$.

Corollary 6.4.4. *Let N be a collection of subsets over a base set U and let $C \subset U$ be the optimal Jaccard center. Assuming ETH, no FPT algorithm with parameter $k = \max_{X \in N} |C \triangle X|$, can run in time $2^{o(k)}$. Further, no PTAS for the Jaccard Center problem can run in time $2^{o(\sqrt{1/\varepsilon})}$.*

Proof. Under ETH, no FPT algorithm for vertex cover with parameter $|K|$, the minimal size of the vertex cover, can run in time $2^{o(|K|)}$, see Cai and Juedes [82]. Since $k = \max_{X \in N} |C \triangle X| \in \Theta(|K|)$, the first claim follows. For the second claim, recall any PTAS approximating the Jaccard center problem beyond a factor of $(1 + \frac{1}{|U|^2})$ recovers the optimal solution. □

6.5 Core-Covers

Our algorithms are based on the existence of a small collection M of input sets such that a high-quality center can be extracted from M. Informally, the items of an optimal center are well represented by the items of the sets contained in M. The construction is somewhat inspired by coresets for the Euclidean minimum enclosing ball problem, albeit with a weaker guarantee.

Definition 6.5.1 (Core-Covers)**.** Let N be a collection of subsets of a base set U, let OPT be the maximum distance of an optimal Jaccard center to any subset in N, and let $\alpha \geq 1$ be a parameter. A collection $M \subseteq N$ is called an *α-core-cover* if there exists an optimal center C with $K = C \cap \left(\bigcup_{X \in M} X \right)$ and

$$\max_{X \in N} D(X, K) \leq \alpha \cdot \mathrm{OPT}.$$

A collection $M \subseteq N$ with $A_M = \bigcap_{X \in M} X$ and $O_M = \bigcup_{X,Y \in M} X \triangle Y$ is called an *anchored α-core-cover* if there exists an optimal center C with

$$\max_{X \in N} D(X, A_M \cup (O_M \cap C)) \leq \alpha \cdot \mathrm{OPT}.$$

We are especially interested in the size of core-covers with $\alpha = 1$ or $\alpha = 1 + \varepsilon$. Core-covers are useful when the supports, i.e. the cardinalities of the sets X, are small, in which case we can find the solution by enumerating over all possible subsets of $\bigcup_{X \in M} X$. Anchored core-covers are more convenient if the supports are large while the optimum value is small. For the remainder of this section, we will give (non-constructive) upper and lower bounds on the number of points required to satisfy both guarantees. Our proofs are essentially based on the following observation.

Observation 6.5.2. *For any three sets $C, K, X \subseteq U$*

$$D(X, K) \leq D(X, K \cap C) + \frac{|K \setminus C| - 2|(X \cap K) \setminus C)|}{|X \cup K|}.$$

Proof.

$$\begin{aligned} D(X,K) &= \frac{|X \triangle K|}{|X \cup K|} = \frac{|X \triangle (K \cap C)| + |(K \setminus C) \setminus X| - |X \cap (K \setminus C)|}{|X \cup (K \cap C)| + |K \setminus C \setminus X|} \\ &\leq \frac{|X \triangle (K \cap C)|}{|X \cup (K \cap C)|} + \frac{|(K \setminus C) \setminus X| - |X \cap (K \setminus C)|}{|X \cup K|} \\ &= D(X, K \cap C) + \frac{|K \setminus C| - 2|X \cap (K \setminus C)|}{|X \cup K|} \end{aligned}$$

□

If X is an arbitrary input point, K is our possible solution, and C is an optimal center, this observation implies that it is sufficient to show that $D(X, K \cap C)$ is a good approximation to $D(X, C)$ and $\frac{|K \setminus C| - 2|(X \cap K) \setminus C)|}{|X \cup K|}$ is small or negative.

Lemma 6.5.3. *For any collection of subsets N, there exists an α-core-cover M of size $\lceil 1/\varepsilon \rceil + 1$ if $\alpha = 1 + \varepsilon$ with $\varepsilon > 0$ and $\min\left\{\frac{\log(OPT \cdot |C|)}{\log(2 - OPT)} + 1, |C|\right\}$ if $\alpha = 1$.*

Proof. We show the existence of the collection M by proving that we can iteratively add a set to M such that either K is already a good approximate solution or the added set contains many elements from $C \setminus K$. Thus, finally we either have C covered by $\bigcup_{X \in M} X$ or no set violates the approximation guarantee. Let $M^{(0)} = \{X\}$ for an arbitrary $X \in N$. We denote by $K^{(i)} = C \cap \left(\bigcup_{X \in M^{(i)}} X\right)$ our solution after the i-th iteration. Note that due to Fact 6.3.1, we can assume $|C \setminus K^{(i)}| \leq \mathrm{OPT} \cdot |C|$ as $M^{(i)}$ is non-empty.

We first consider the case $\alpha = 1 + \varepsilon$. If $\mathrm{OPT} \geq \frac{1}{1+\varepsilon}$, then any set is a $(1+\varepsilon)$-approximation as $D(X, Y) \leq 1$ for any two item set $X, Y \subseteq U$. This in particular implies that there exists an $(1 + \varepsilon)$-cover of size 1.

Now let $\mathrm{OPT} < \frac{1}{1+\varepsilon}$ which implies $1 - (1 + \varepsilon) \cdot \mathrm{OPT} > 0$. Further let $X \in N$ be a set such that $D\left(X, K^{(i)}\right) > (1 + \varepsilon) \cdot \mathrm{OPT}$. Then

$$
\begin{aligned}
|X \cap (C \setminus K^{(i)})| \overset{K^{(i)} \subseteq C}{=} & |X \cap C| - |X \cap K^{(i)}| \\
\geq & (1 - \mathrm{OPT}) \cdot |X \cup C| - (1 - D(X, K^{(i)})) \cdot |X \cup K^{(i)}| \\
= & (1 - \mathrm{OPT}) \cdot |X \cup C| - \\
& (1 - D(X, K^{(i)})) \cdot (|X \cup C| - |C \setminus K^{(i)}| + |X \cap (C \setminus K^{(i)})|) \\
> & (1 - \mathrm{OPT}) \cdot |X \cup C| - \\
& (1 - (1 + \varepsilon) \cdot \mathrm{OPT}) \cdot (|X \cup C| - |C \setminus K^{(i)}| + |X \cap (C \setminus K^{(i)})|) \\
= & \varepsilon \cdot \mathrm{OPT} \cdot |C| + (1 - (1 + \varepsilon) \cdot \mathrm{OPT}) \cdot (|C \setminus K^{(i)}| - |X \cap (C \setminus K^{(i)})|) \\
\geq & \varepsilon \cdot \mathrm{OPT} \cdot |C|,
\end{aligned}
$$

where the first inequality follows from Fact 6.3.1, the second equality follows due to the identity $|X \cup K^{(i)}| = |X \cup C| - |C \setminus K^{(i)}| + |X \cap (C \setminus K^{(i)})|$, and the second inequality due to our choice of X and $1 - (1 + \varepsilon) \cdot \mathrm{OPT} > 0$. Since $|C \setminus K^{(0)}| \leq \mathrm{OPT} \cdot |C|$, after adding at most $s = \lceil 1/\varepsilon \rceil$ sets to $M^{(0)}$, we have $K^{(s)} = C$, or no set X with $D\left(X, K^{(s)}\right) > (1 + \varepsilon) \cdot \mathrm{OPT}$ exists.

Now let us consider the case $\alpha = 1$. We require the following proposition, which will also be used later in Section 6.7

Proposition 6.5.4. *Let $X \in N$, C a set with $D(X,C) \leq OPT < 1$, and $K \subset C$ be a set such that $D\left(X, K^{(i)}\right) > OPT$. Then*

$$|X \cap (C \setminus K)| \geq \frac{1 - OPT}{2 - OPT} \cdot |C \setminus K|.$$

Proof. Let $X \in N$ be a set such that $D\left(X, K^{(i)}\right) > \text{OPT}$. Then

$$\begin{aligned}
|X \cap (C \setminus K)| &\overset{K \subseteq C}{=} |X \cap C| - |X \cap K| \\
&\geq (1 - \text{OPT}) \cdot |X \cup C| - (1 - D(X,K)) \cdot |X \cup K| \\
&= (1 - \text{OPT}) \cdot |X \cup C| - \\
&\quad (1 - D(X,K)) \cdot (|X \cup C| - |C \setminus K| + |X \cap (C \setminus K)|) \\
&> (1 - \text{OPT}) \cdot |X \cup C| - \\
&\quad (1 - \text{OPT}) \cdot (|X \cup C| - |C \setminus K| + |X \cap (C \setminus K)|) \\
&= (1 - \text{OPT}) \cdot (|C \setminus K| - |X \cap (C \setminus K)|),
\end{aligned}$$

where the first inequality follows from Fact 6.3.1, the second equality follows due to the identity $|X \cup K| = |X \cup C| - |C \setminus K| + |X \cap (C \setminus K)|$, and the second inequality due to our choice of X. Rearranging terms of the final inequality yields the claim. □

The proposition implies that X covers at least $\frac{1-\text{OPT}}{2-\text{OPT}}$ items from $C \setminus K^{(i))}$ in iteration i. Thus, $|C \setminus K^{(i)}| \leq (1 - \frac{1-\text{OPT}}{2-\text{OPT}})^i |C \setminus K^{(0)}| \leq (\frac{1}{2-\text{OPT}})^i \cdot \text{OPT} \cdot |C|$ which is smaller than 1 if $i > \frac{\log(\text{OPT} \cdot |C|)}{\log(2-\text{OPT})}$. Note that $|X \cap (C \setminus K^{(i)})| \geq 1$ if $D(X, K^{(i)}) > OPT$ which concludes the proof. □

With the space bound for core-covers, we can prove the main result of this section.

Lemma 6.5.5. *For any collection of subsets N, there exists an anchored α-core-cover $M \subset N$ of size $O(1/\varepsilon)$ if $\alpha = 1 + \varepsilon$ with $\varepsilon > 0$ and of size $\min\{\frac{\log(OPT \cdot |C|)}{\log(2-OPT)} + 1, |C|\} + \log \frac{OPT \cdot |C|}{1-OPT}$ if $\alpha = 1$.*

Proof. Assume we have some optimal center C. Lemma 6.5.3 gives a set M such that $K \cap C$ is an α-approximate solution where we can represent K as $K = A_M \cup (O_M \cap C)$. Using Observation 6.5.2, the distance between K and some arbitrary set X is

$$\begin{aligned}
D(X,K) &\leq D(X, K \cap C) + \frac{|K \setminus C| - 2 \cdot |(X \cap K) \setminus C)|}{|X \cup K|} \\
&= D(X, K \cap C) + \frac{|A_M \setminus C| - 2 \cdot |X \cap (A_M \setminus C)|}{|X \cup K|} \\
&\leq \alpha \cdot \text{OPT} + \frac{|A_M \setminus C| - 2 \cdot |X \cap (A_M \setminus C)|}{|X \cup K|}.
\end{aligned}$$

If for every $X \in N$, we have $2 \cdot |X \cap (A_M \setminus C)| > |A_M \setminus C|$ then the ratio is negative and $D(X,K) \leq D(X, K \cap C) \leq \alpha \cdot \mathrm{OPT}$. Otherwise, there exists an X such that $|(X \cap A_M) \setminus C| = |X \cap (A_M \setminus C)| \leq |A_M \setminus C|/2$. We iteratively augment the collection M satisfying the space and approximation bounds of Lemma 6.5.3 with additional sets X. In each iteration, $|A_M \setminus C|$ is halved.

If $\alpha = 1$ and after adding $i > \log |A_M \setminus C|$ sets, we have $A_M \setminus C = \emptyset$. For a more precise bound on i let $Y \in M$. Then due to Fact 6.3.1,

$$|A_M \setminus C| \leq |Y \setminus C| \leq \mathrm{OPT} \cdot |Y| \leq \mathrm{OPT} \cdot |Y \cup C| \leq \frac{\mathrm{OPT} \cdot |Y \cap C|}{1 - \mathrm{OPT}} \leq \frac{\mathrm{OPT} \cdot |C|}{1 - \mathrm{OPT}}. \quad (6.1)$$

For the case $\alpha = 1 + \varepsilon$, let us first consider $\mathrm{OPT} \geq \frac{1}{1+\varepsilon}$. Then any point is a $(1+\varepsilon)$-approximation which also implies that there exists an anchored core cover of size 1. Now let $\mathrm{OPT} < 1/(1+\varepsilon)$, $X \in N$, and assume that we are given an $\alpha' = (1 + \varepsilon/2)$-core-cover. Again due to Fact 6.3.1 and using the same derivation as for Equation 6.1, we have

$$\begin{aligned} |A_M \setminus C| &\leq \mathrm{OPT} \cdot \frac{|C|}{1 - \mathrm{OPT}} \leq \mathrm{OPT} \cdot \frac{|X|}{(1 - \mathrm{OPT})(1 - D(X,C))} \leq \mathrm{OPT} \cdot \frac{|X|}{(1 - \mathrm{OPT})^2} \\ &\leq \mathrm{OPT} \cdot \frac{(1+\varepsilon)^2 \cdot |X|}{\varepsilon^2} \leq \mathrm{OPT} \cdot \frac{4}{\varepsilon^2} \cdot |X|, \end{aligned}$$

where the last inequality follows for $\varepsilon \leq 1$. After adding $\log \frac{8}{\varepsilon^3}$ sets such that $|A_M \setminus C|$ is halved with each sets, we have $|A_M \setminus C|/|X \cup K| \leq \varepsilon \cdot \mathrm{OPT} \cdot |X|/|X \cup K| \leq \varepsilon/2 \cdot \mathrm{OPT}$. Our approximation factor is therefore $\alpha' \cdot \mathrm{OPT} + \varepsilon/2 \cdot \mathrm{OPT} = (1+\varepsilon) \cdot \mathrm{OPT}$. □

We would like to remark that the bound on the number of sets required to satisfy the $(1+\varepsilon)$-core-cover guarantee is tight, and that the bound on the number of sets to satisfy the anchored $(1+\varepsilon)$-core-cover guarantee is tight up to constant multiplicative factors. Note that M is constrained to using only input sets. Better bounds are possible when we lift this restriction on M (for instance, if M consists of only an optimum center C then all guarantees are met). It is unclear whether improved guarantees not using input sets can be feasibly used in an algorithm.

Lemma 6.5.6. *There exists a collection of subsets N such that for any $M \subseteq$ satisfying the $(1+\varepsilon)$-core-cover or anchored $(1+\varepsilon)$-core-cover guarantee, we have $|M| \geq 1/\varepsilon - 1$.*

Proof. For a given $\varepsilon > 0$ and assuming $1/\varepsilon$ to be an integer, we consider the following instance of vertex cover. We are given $1/\varepsilon - 1$ stars, each with at least two leaves. The optimum vertex cover and the optimum Jaccard center consists of the internal nodes, with an optimum objective value for the Jaccard center of $\frac{1/\varepsilon - 1}{1/\varepsilon}$. If M does not consist of at least one edge from each star, corresponding to a set containing the element contained in the optimal Jaccard center, any center computed using only the entries of the picked edges will not intersect with at least one star, i.e. have distance 1 to the edges of the omitted star. Since

$\frac{1/\varepsilon-1}{1/\varepsilon} \cdot (1+\varepsilon) = 1-\varepsilon^2 < 1$, M has to hit every star, i.e. has to consist of at least $1/\varepsilon - 1$ edges. □

6.6 A PTAS for Binary Jaccard Center

This section mainly consists of the proof of the following theorem.

Theorem 6.6.1. *Given a collection N of n subsets from a base set U and any $\varepsilon > 0$, there exists an algorithm computing a $(1+\varepsilon)$-approximation to the optimal Jaccard center with probability at least $1/2$. The algorithm runs in time $|U|^2 \cdot (n^{O(\varepsilon^{-6})} + LP(n,|U|))$, where $LP(n,|U|)$ is the time required to solve a linear program with n constraints and $|U|$ variables.*

The algorithm (see also Algorithm 8) consists of two main steps. Let OPT be the optimal objective value. Since there are $O(|U|^2)$ distinct objective values for the Jaccard center problem with a base set of size d, we can try to a find solution for each value (c.f. line 3 Algorithm 8). Recall that $C_i = \begin{cases} 0 & \text{if } i \notin C \\ 1 & \text{if } i \in C \end{cases}$ and that $D(X,C) \leq \text{OPT}$ holds for all $X \in N$. By multiplying both sides of the inequality with $|X \cup C|$, we obtain

$$|X \triangle C| \leq \widehat{\text{OPT}} \cdot |X \cup C|. \tag{6.2}$$

Observe that $|X \triangle C| = \sum_{i=1}^{|U|} X_i - 2X_iC_i + C_i$ and $|X \cup C| = \sum_{i=1}^{|U|} X_i - X_iC_i + C_i$. Hence, we obtain a set of linear inequalities which we can test for feasibility by relaxing the integrality constraints on C. Denote a feasible non-integral solution by C'. The existence of a feasible integral solution of Equation 6.2 implies a feasible relaxed solution C'. We interpret the C'_i as probabilities, i.e. we obtain a binary vector C by rounding each C'_i to 1 with probability C'_i. Using Chernoff bounds, this approach yields a good solution if $\text{OPT} \cdot |X| > s \cdot \log n/\varepsilon^2$ for all X and some constant s (c.f. lines 4-7 of Algorithm 8).

If $\text{OPT} \cdot |Y|$ is smaller than this threshold for at least one $Y \in N$ then we could employ a naive brute force algorithm by iterating over all $\binom{|U|}{s \cdot \log n/\varepsilon} \in O(|U|^{s \cdot \log n/\varepsilon})$ subsets S and outputting the best $Y \triangle S$. To eliminate the dependency on $|U|$, we first show that a bound on $\text{OPT} \cdot |Y|$ implies that $|X_1 \triangle X_2|$ for any two sets $X_1, X_2 \in N$ is bounded. Then we compute an anchored core-cover M by enumerating all collections of $O(1/\varepsilon)$ input sets. Having determined M, computing the optimum $A_M \cup S$ with $S \subseteq O_M$ becomes feasible (c.f. lines 9-11 of Algorithm 8).

Proof of Theorem 6.6.1. In the following, we always assume that $\text{OPT} < 1/(1+\varepsilon)$, as otherwise any solution is a $(1+\varepsilon)$ approximation.

To round the set of linear Equations 6.2, we first apply Chernoff Bounds (Theorem 2.1.5).

Lemma 6.6.2. *Let S be a random binary vector obtained by rounding a fractional feasible solution of the set of Equations 6.2 and let $\varepsilon > 0$ be a constant. Assume that $OPT \cdot |X| \geq \frac{27\ln(4n)}{\varepsilon^2}$ for all $X \in N$. Then with probability at least $1/2$, the rounding procedure produces a binary solution S with $\max_{X \in N} D(X,S) \leq (1+\varepsilon) \cdot OPT$.*

Proof. Observe that $\mathbb{E}[|X \cup S|] \geq |X|$. We first derive concentration bounds on $|X \triangle S|$ and $|X \cup S|$. For any $X \in N$, Theorem 2.1.5 yields

$$\mathbb{P}\left[|X \cup S| < (1-\varepsilon/3) \cdot \mathbb{E}[|X \cup S|]\right] \leq \exp\left(-\frac{\varepsilon^2 \cdot \mathbb{E}[|X \cup S|]}{18}\right) \leq \exp\left(-\frac{\varepsilon^2 \cdot |X|}{18}\right) \leq \frac{1}{4n}$$

and

$$\begin{aligned}
& \mathbb{P}\left[|X \triangle S| > \mathbb{E}[|X \triangle S|] + \varepsilon/3 \cdot \mathrm{OPT} \cdot \mathbb{E}[|X \cup S|]\right] \\
= \; & \mathbb{P}\left[|X \triangle S| > \left(1 + \frac{\varepsilon \cdot \mathrm{OPT} \cdot \mathbb{E}[|X \cup S|]}{3 \cdot \mathbb{E}[|X \triangle S|]}\right) \cdot \mathbb{E}[|X \triangle S|]\right] \\
\leq \; & \exp\left(-\frac{\varepsilon^2 \cdot \mathrm{OPT}^2 \cdot \mathbb{E}[|X \cup S|]^2}{27 \cdot \mathbb{E}[|X \triangle S|]^2} \cdot \mathbb{E}[|X \triangle S|]\right) \\
\leq \; & \exp\left(-\varepsilon^2 \cdot \mathrm{OPT} \cdot \mathbb{E}[|X \cup S|]/27\right) \leq \exp\left(-\varepsilon^2 \cdot \mathrm{OPT} \cdot |X|/27\right) \leq \frac{1}{4n}.
\end{aligned}$$

Combining these two bounds, we have

$$\frac{|X \triangle S|}{|X \cup S|} \leq \frac{\mathbb{E}[|X \triangle S|] + \varepsilon/3 \cdot \mathrm{OPT} \cdot \mathbb{E}[|X \cup S|]}{(1-\varepsilon/3) \cdot \mathbb{E}[|X \cup S|]} \leq \frac{\mathrm{OPT} + \varepsilon/3 \cdot \mathrm{OPT}}{1-\varepsilon/3} \leq (1+\varepsilon) \cdot \mathrm{OPT}$$

with probability at least $1 - 1/2n$. Applying the union bound, we then obtain

$$\begin{aligned}
& \mathbb{P}\left[\max_{X \in N} D(X,S) \leq (1+\varepsilon) \cdot \mathrm{OPT}\right] \\
= \; & 1 - \mathbb{P}\left[\exists X \in N : \; \frac{|X \triangle S|}{|X \cup S|} > (1+\varepsilon) \cdot \mathrm{OPT}\right] \geq 1 - \frac{n}{2n} = 1/2.
\end{aligned}$$

□

If $\mathrm{OPT} \cdot |X| > \frac{27\ln(4n)}{\varepsilon^2}$ for all $X \in N$, we can use the LP-based rounding scheme analyzed in Lemma 6.6.2 (c.f. lines 4-7 of Algorithm 8). For the other cases, we will utilize Lemma 6.5.5 as follows. There exists at least one set Y with $\mathrm{OPT} \cdot |Y| \leq \frac{27\ln(4n)}{\varepsilon^2}$. With Fact 6.3.1, we have $\mathrm{OPT} \cdot |C| \leq \mathrm{OPT} \cdot |Y|/(1 - D(Y,C)) \leq \mathrm{OPT} \cdot |Y|/(1-\mathrm{OPT}) \leq \frac{27 \cdot (1+\varepsilon) \cdot \ln(4n)}{\varepsilon^3}$. For any two sets $X_1, X_2 \in N$, we then have

$$\begin{aligned}
|X_1 \triangle X_2| &\leq 2 \cdot \mathrm{OPT} \cdot |X_1 \cup X_2| \leq 2 \cdot \mathrm{OPT} \cdot (|X_1| + |X_2|) \\
&\leq 4 \cdot \mathrm{OPT} \frac{|C|}{1-\mathrm{OPT}} \leq \frac{108 \cdot (1+\varepsilon)^2 \cdot \ln(4n)}{\varepsilon^4}.
\end{aligned}$$

Algorithm 8 PTAS for the Jaccard center problem

Let $D = \{\frac{i}{j} \mid 1 \leq j \leq d \text{ and } 0 \leq i < j\}$
Initialize list $C = \emptyset$
for all $\widehat{\mathrm{OPT}} \in D$ **do**
  if $\exists X \in N \,.\, \widehat{\mathrm{OPT}} \cdot X < \frac{27 \ln(4n)}{\varepsilon^2}$ **then**
    for all $M \subseteq N$ with $|M| = \lceil \frac{5}{\varepsilon} + 5 \rceil$ **do**
      Compute optimal solution $K_{\widehat{\mathrm{OPT}}} = A_M \cup S$ with $S \subseteq O_M$ (cf. Lemma 6.5.5)
      Add $K_{\widehat{\mathrm{OPT}}}$ to C
    end for
  else
    Obtain non-integral solution $K'_{\widehat{\mathrm{OPT}}}$ by solving the set of linear equations given by 6.2
    Obtain $K_{\widehat{\mathrm{OPT}}}$ by rounding each entry of $K'_{\widehat{\mathrm{OPT}}}$
    Add $K_{\widehat{\mathrm{OPT}}}$ to C
  end if
end for
return $\min_{\widehat{\mathrm{OPT}} \in D} \{K_{\widehat{\mathrm{OPT}}} \in C\}$

Let M now be a collection of sets satisfying the anchored $(1+\varepsilon)$-core-cover guarantee of Lemma 6.5.5 with $A_M = \bigcap_{X \in M} X$ and $O_M = \bigcup_{X,Y \in M} X \triangle Y$. Such a collection can be determined in time $n^{O(\varepsilon^{-1})}$ by iterating through all subsets of N of cardinality $O(\varepsilon^{-1})$. Since $|O_M| \leq \sum_{X_i \in M} \sum_{X_j \in M} |X_i \triangle X_j| \leq |M|^2 \cdot \max_{X_i, X_j \in M} |X_i \triangle X_j| \in O(\log n \cdot \varepsilon^{-6})$, we can compute an optimal solution of $\max_{X \in N} \min_{S \subseteq O_M} D(X, A_M \cup S)$ in time $2^{|O_M|} = 2^{O(\log n \cdot \varepsilon^{-6})}$.

The total running time amounts to $|U|^2$ calls to the LP given via Equations 6.2 or $|U|^2$ applications of Lemma 6.5.5 with a running time of $2^{O(\log n \cdot \varepsilon^{-6})} = n^{O(\varepsilon^{-6})}$. □

6.7 An FPT Algorithm for Binary Jaccard Center

Our second application of core-covers is an FPT algorithm in the parameter $k = \max_{X \in N} |X \triangle C|$ where C is an arbitrary optimal solution. The main technical difficulty is to efficiently construct a core-cover without enumerating all possible core-covers.

We first bound the size of an anchored 1-core-cover given by Lemma 6.5.5 in terms of k.

Lemma 6.7.1. *For any collection N of subsets and an optimal center C with cost $OPT < 1$, let $k = \max_{X \in N} |X \triangle C| > 2$. Then*

$$\min\left\{\frac{\log(OPT \cdot |C|)}{\log(2 - OPT)} + 1, |C|\right\} \leq 2k \text{ and } \log \frac{OPT \cdot |C|}{1 - OPT} \leq 3 \log k.$$

Proof. There exists an $X \in N$ such that $k \geq |X \triangle C| = \mathrm{OPT} \cdot |X \cup C| \geq \mathrm{OPT} \cdot |C|$. If

$\mathrm{OPT} \leq 1/2$ then

$$\min\left\{\frac{\log(\mathrm{OPT}\cdot|C|)}{\log(2-\mathrm{OPT})}+1,|C|\right\} \leq 1+2\log k \text{ and } \log\frac{\mathrm{OPT}\cdot|C|}{1-\mathrm{OPT}} \leq \log k.$$

If $1 > \mathrm{OPT} > 1/2$ then we have $|X \triangle C|/|X \cup C| = \mathrm{OPT}$ for some $X \in N$ implying

$$(1-\mathrm{OPT}) = \frac{|X\cap C|}{|X\cup C|} = \mathrm{OPT}\cdot\frac{|X\cap C|}{|X\triangle C|} \geq \frac{1}{2|X\triangle C|} \geq \frac{1}{2k}.$$

Therefore, we have $1/(1-\mathrm{OPT})) \leq 2k$,

$$\log(2-\mathrm{OPT}) = \log(1+1-\mathrm{OPT}) = \frac{\ln(1+1-\mathrm{OPT})}{\ln 2} \geq \frac{1-\mathrm{OPT}}{2\ln 2} \geq \frac{1}{4k\ln 2},$$

and

$$\min\left\{\frac{\log(\mathrm{OPT}\cdot|C|)}{\log(2-\mathrm{OPT})}+1,|C|\right\} \leq |C| \leq 2k \text{ and } \log\frac{\mathrm{OPT}\cdot|C|}{1-\mathrm{OPT}} \leq 2\log 2k < 3\log k).$$

□

Algorithm 9 FPT-algorithm for the Jaccard center problem

```
Let D = {i/j | 1 ≤ j ≤ d and 0 ≤ i < j}
Initialize list C = ∅
for all ÔPT ∈ D do
    Initialize M = {X, Y} with arbitrary X, Y ∈ N and X ≠ Y
    for i = 1 to 9k log k do
        Compute optimal solution K_ÔPT = A_M ∪ S with S ⊆ O_M (cf. Lemma 6.5.5)
        if ∃X ∈ N : D(X, K_ÔPT) > ÔPT then
            M = M ∪ {X}
        else
            Add K_ÔPT to C
            break
        end if
    end for
end for
return min_{ÔPT∈D} {K_ÔPT ∈ C}
```

For a given estimate of OPT, the algorithm initially chooses two arbitrary sets to be included in the anchored core-cover M. If the optimal solution $A_M \cup S$ with $S \subseteq O_M$ satisfies $\max_{X\in N} D(X, A_M \cup S) < \mathrm{OPT}$ then we can reduce our estimate of OPT. Otherwise, we add any set X at distance greater than OPT to M. The set X improves the core-cover, either by increasing $|C \cap (A_M \cup O_M)|$ or by decreasing $|A_M \setminus C|$ for some optimal center C. Lemma 6.7.1 allows us to bound the number of times this happens before M satisfies the

anchored core-cover guarantee, upon which we can recover the optimum solution.

Theorem 6.7.2. *Algorithm 9 computes an optimal Jaccard center C satisfying* $\max_{X\in N} |X \triangle C| = k$ *in time* $2^{O(k^3)} \cdot n \cdot |U|^3$.

Proof. Let $\widehat{\mathrm{OPT}} \in D$ be a guess for our optimal value OPT. If $\widehat{\mathrm{OPT}} < \mathrm{OPT}$ then the loop terminates without finding a center. Let $\widehat{\mathrm{OPT}} \geq \mathrm{OPT}$. Using Observation 6.5.2, we know that

$$D(X, K_{\widehat{\mathrm{OPT}}}) \leq D(X, K_{\widehat{\mathrm{OPT}}} \cap C) + \frac{|K_{\widehat{\mathrm{OPT}}} \setminus C| - 2 \cdot |(X \cap K_{\widehat{\mathrm{OPT}}}) \setminus C)|}{|X \cup K_{\widehat{\mathrm{OPT}}}|}.$$

If $D(X, K_{\widehat{\mathrm{OPT}}}) > \widehat{\mathrm{OPT}}$ then we distinguish between two cases:

Case $|K_{\widehat{\mathrm{OPT}}} \setminus C| - 2 \cdot |(X \cap K_{\widehat{\mathrm{OPT}}}) \setminus C)| \leq 0$
Then $D(X, K_{\widehat{\mathrm{OPT}}} \cap C) > \widehat{\mathrm{OPT}}$. Using Proposition 6.7, we have $|X \cap (C \setminus K_{\widehat{\mathrm{OPT}}})| > \frac{1-\mathrm{OPT}}{2-\mathrm{OPT}} \cdot |C \setminus K_{\widehat{\mathrm{OPT}}}| > 0$. Thus, by adding X to M, $|C \setminus K_{\widehat{\mathrm{OPT}}}|$ is reduced by a factor of $(1 - \frac{1-\mathrm{OPT}}{2-\mathrm{OPT}}) = \frac{1}{2-\mathrm{OPT}}$. Since initially some item $Y \in M$, we have $|C \setminus K_{\widehat{\mathrm{OPT}}}| \leq |C \setminus Y| \leq \mathrm{OPT} \cdot |C|$ due to Fact 6.3.1. Therefore we can add at most $1 + \log \frac{\mathrm{OPT} \cdot |C|}{2-\mathrm{OPT}}$ items X before $C \setminus K_{\widehat{\mathrm{OPT}}}$ is empty. Additionally, we can add at most $|C|$ items before $C \setminus K_{\widehat{\mathrm{OPT}}}$ is empty. Combining this with Lemma 6.7.1, then shows that we add at most $2k$ sets to M until this case can no longer occur.

Case $|K_{\widehat{\mathrm{OPT}}} \setminus C| - 2 \cdot |(X \cap K_{\widehat{\mathrm{OPT}}}) \setminus C)| > 0$
Then $|(X \cap K_{\widehat{\mathrm{OPT}}}) \setminus C)| \leq |K_{\widehat{\mathrm{OPT}}} \setminus C|/2$. If we add $i > \log |A_M \setminus C|$ sets to M, we have $K_{\widehat{\mathrm{OPT}}} \setminus C = \emptyset$. Since initially some set $Y \in M$, we have due to Fact 6.3.1, we have

$$|A_M \setminus C| \leq |Y \setminus C| \leq \mathrm{OPT} \cdot |Y| \leq \mathrm{OPT} \cdot |Y \cup C| \leq \frac{\mathrm{OPT} \cdot |Y \cap C|}{1 - \mathrm{OPT}} \leq \frac{\mathrm{OPT} \cdot |C|}{1 - \mathrm{OPT}}.$$

Due to Lemma 6.7.1 this case can only occur $3 \log k$ times.

The computation of $K_{\widehat{\mathrm{OPT}}}$ can be done in time $2^{O(k^3)}$ by an exhaustive search over all possible subsets of O_M since

$$|O_M| \leq |M^2| \cdot \max_{X,Y \in N} |X \triangle Y| \leq O(k^2) \cdot \max_{X,Y \in N} (|X \triangle C| + |Y \triangle C|) = O(k^3).$$

We perform the exhaustive search $2k + 3 \log k \in O(k)$ times and for each solution we evaluate the objective value for each set. Since $|D| = O(|U|^2)$ and we examine every set in line 7 of Algorithm 9, the algorithm terminates in time $2^{O(k^3)} \cdot n \cdot |U|^3$. □

6.8 A Note on Continuous Jaccard Center

In this section we briefly remark on how to find the Jaccard center for non-negative real vectors. For such inputs, the Jaccard measure is defined as follows.

Definition 6.8.1 (Continuous Jaccard Measures)**.** Given two d dimensional vectors X, Y with non-negative real entries, the *continuous Jaccard similarity* is defined as

$$J(X,Y) = \begin{cases} \frac{\sum_{i=1}^{d} \min(X_i,Y_i)}{\sum_{i=1}^{d} \max(X_i,Y_i)} & \text{if } \sum_{i=1}^{n} \max(X_i,Y_i) > 0 \\ 1 & \text{if } \sum_{i=1}^{d} \max(X_i,Y_i) = 0, \end{cases}$$

and the *continuous Jaccard distance* is defined as $D(X,Y) = 1 - J(X,Y)$.

We will formulate the decision problem of finding a center with distance at most *dist* as an LP. The optimum center can thereafter be determined in polynomial time using binary search over the possible values of *dist*. In the following let $X^j \in N$ be the jth point of N w.r.t. some arbitrary ordering. We will use the variable $c_i \geq 0$ to denote the ith entry of the Jaccard center. We further use the variables $a_{i,j}$ and $b_{i,j}$ for all $i \in \{1, \ldots, d\}$ and $j \in \{1, \ldots n\}$ to denote the maximum and minimum of X_i^j and c_i. We then use the following constraints for all points X^j

$$\begin{aligned} \sum_{i=1}^{n} b_{i,j} \geq (1 - dist) \cdot \sum_{i=1}^{n} a_{i,j} & \\ b_{i,j} \leq c_i, X_i^j & \qquad \text{for all } i \in [n], j \in [m] \\ a_{i,j} \geq c_i, X_i^j & \qquad \text{for all } i \in [n], j \in [m] \\ a_{i,j}, b_{i,j}, c_i \geq 0 & \qquad \text{for all } i \in [n], j \in [m] \end{aligned}$$

Note that the top most equation corresponds to $\sum_{i=1}^{n} \min(c_i, X_i^j) \geq (1 - dist) \cdot \sum_{i=1}^{n} \sum_{i=1}^{n} \max(c_i, X_i^j)$ which is equal to $1 - \frac{\sum_{i=1}^{n} \min(X_i,Y_i)}{\sum_{i=1}^{n} \max(X_i,Y_i)} \leq dist$.

7 A Benchmark Algorithm for Clustering? The Success of Local Search

As described in Chapter 1, most center based clustering tasks are well, if not completely understood from a worst-case perspective. For most relevant metrics, we either have APX-hardness results, often with matching, or near matching approximation ratios, or polynomial time approximation schemes. While there exist problems with a worse worst-case behavior, these results would typically rank clustering among the more difficult problems.

With this in mind, we might find it surprising that practitioners generally regard clustering to be "easy". From a practitioner's point of view, the appropriateness of a particular clustering objective depends on the underlying structure of the data. The main focus is less on improving a computationally difficult task, but to identify structural properties and correctly model the data. Given an appropriate model, we can often expect a benchmark algorithm to yield a good clustering. This gives clustering its easy-in-practice, hard-in-theory quality. To bridge this gap, prior work usually proceeds in two steps: (1) characterize properties of a natural clustering of the underlying data and (2) design an algorithm leveraging such properties, which then bypasses traditional hardness results. At this point, there is a wide variety of characterizations of well-behaved instances and of algorithms tuned to those instances.

In contrast, we proceed in the reverse order: (1) focus on a single, all-purpose algorithm that is already widely used in practice, and (2) prove that is works well for most models of well-clusterable instances for the k-median and k-means objective functions. The algorithm: a simple Local Search heuristic.

This yields a unified and simple approach toward stability conditions. There are two possible high-level interpretations of our results: (1) since Local Search heuristics are widely used by practitioners, our work shows that the three main stability conditions capture some of the structure of practical inputs that make Local Search efficient, giving more legitimacy to the stability conditions and (2) assuming that the stability conditions are legitimate (*i.e.*: characterize real-world instances), our results make a step toward understanding the success of Local Search heuristics.

We now proceed to a more formal exposition of our contribution. The problem we consider in this work is (k, p) clustering for constant values of p, see Definition 2.5.1, such as k-median and k-means clustering. We will analyze the performance of the following widely-used Local Search algorithm (Algorithm 10) (see e.g.: [1] or [208]). This algorithm has a polynomial

running time if the inital solution is an $\exp(n)$ approximation (see [30, 112]). In the following we will refer to its parameter ((ε^{-1}) in the description of Algorithm 10) as the *neighborhood size* of Local Search.

Algorithm 10 Local Search(ε^{-1}) for k-Median and k-Means

1: **Input:** A, F, cost, k
2: $S \leftarrow$ Arbitrary subset of F of cardinality at most k.
3: **while** $\exists\ S'$ s.t. $|S'| \leq k$ **and** $|S \setminus S'| + |S' \setminus S| \leq \varepsilon^{-1}$ **and** $\text{cost}(S') \leq (1 - \varepsilon/n)\ \text{cost}(S)$
4: **do**
5: $\quad S \leftarrow S'$
6: **end while**
7: **Output:** S

Our Contribution Several approaches have been proposed to bridge the gap between theory and practice. For example, researchers have considered the average-case scenario [58] where the running time of an algorithm is analyzed with respect to some probability distribution over the set of all inputs. Smooth analysis (e. g., [271]) is another celebrated approach that analyzes the running time of an algorithm with respect to worst-case inputs subject to small random perturbations.

Another successful approach, the one we take in this thesis, focuses on *structured* inputs. In a seminal paper, Ostrovsky, Rabani, Schulman, and Swamy [257] introduced the idea that inputs that come from practice have *meaningful* clustering. Specifically, they assumed that a k-means clustering is more meaningful than a $k-1$-means clustering, if the cost is cheaper by at least a constant factor. This is a very strong assumption, and subsequent work further introduced weaker conditions or proposed other stability measures. We will show that Local Search works reasonably well on all of them.

The main message is that Local Search occupies a sweet spot between practical performance and theoretical guarantees, both with respect to worst case instances and with respect to stable instances for various notions of stability. More boldly, our work indicates that many formal characterizations of practical instances can be, at least to some degree, viewed as "instances for which Local Search works well". It supports the definition of stability conditions as conditions characterizing real-world inputs since Local Search heuristics are very popular among practitioners.

While the (worst case) running time bounds given in this work might appear too high for real-world applications, we consider this view as possibly too pessimistic, given that there is a trade-off between how "stable" the instances are and the quality of approximation. Hence if instances are highly stable (*i.e.*: the parameter of the stability condition is high), Local Search *does not* require a large neighborhood size to output a nearly-optimal solution.

The remaining chapter is now organized as follows. We first discuss related work on stability

conditions and Local Search (Section 7.1). The three main, incomparable ones are *distribution stability* [34], analyzed in Section 7.2, *perturbation resilience* [64], analyzed in Section 7.3, and *spectral separability* [222], analyzed in Section 7.4.

7.1 Related Work

There are two general aims that shape the definitions of stability conditions. First, we want the objective function to be appropriate. For instance, if the data is generated by mixture of Gaussians, the k-means objective will be more appropriate than the k-median objective. Secondly, we assume that there exists some ground truth, i.e. a correct assignment of points into clusters. Our objective is to recover this ground truth as well as possible. These aims are not mutually exclusive. For instance, an ideal objective function will allow us to recover the ground truth. We refer to Figure 7.1 for a visual overview of stability conditions and their relationships.

7.1.1 Cost-Based Separation

Given that an algorithm optimized with respect to some objective function, it is natural to define a stability condition as a property the optimum clustering is required to have.

ORSS-Stability [257] Assume that we want to cluster a data set with respect to the k-means objective, but have not decided on the number of clusters. A simple way of determining the "correct" value of k is to run a k-means algorithm for $k \in \{1, 2, \dots m\}$ until the objective value decreases only marginally (using m centers). At this point, we set $k = m - 1$. The reasoning behind this method, commonly known as the *elbow-method* is that we do not gain much information by using m instead of $m-1$ clusters, so we should favor the simpler model. Contrariwise, this implies that we did gain information going from $m - 2$ to $m - 1$ and, in particular, that the $m - 2$-means cost was considerably larger than the $m - 1$-means cost.

Ostrovsky et al. [257] considered whether such discrepancies in the cost also allow us to solve the k-means problem more efficiently, see also Schulman [268] for an earlier condition for two clusters and the irreducibility condition by Kumar et al. [223]. Specifically, they assumed that the optimal k-means clustering has only an ε^2-fraction of the cost of the optimal $(k-1)$-means clustering. For such cost separated instances, the popular D^2-sampling technique has an improved performance compared to the worst-case $O(\log k)$-approximation ratio [27, 70, 200, 257]. Awasthi et al. [34] showed that if an instance is cost-stable, it also admits a PTAS. In fact, they also showed that the weaker condition *β-distribution stability* is sufficient. β-distribution stability states that the cost of assigning a point of cluster C_i to another cluster C_j costs at least β times the total cost divided by the size of cluster C_i. Despite its focus on the properties of the optimum, β-distribution stability has many connections to target-

clustering (see below). Nowadays, the cost-stable property is one of the strongest stability conditions, implying both distribution stability and spectral separability (see below). It is nevertheless the arguably most intuitive stability condition.

Perturbation Resilience The other main optimum-based stability condition is *perturbation resilience*. It was originally considered for the weighted max-cut problem by Bilu et al. [64, 63]. There, the optimum max cut is said to be α-perturbation resilient, if it remains the optimum even if we multiply any edge weight up to a factor of $\alpha > 1$. This notion naturally extends to metric clustering problems, where, given a $n \times n$ distance matrix, the optimum clustering is α-perturbation resilient if it remains optimal if we multiply entries by a factor α. Perturbation resilience has some similarity to smoothed analysis (see Arthur et al. [26, 28] for work on k-means). Both smoothed analysis and perturbation stability aim to study a smaller, more interesting part of the instance space as opposed to worst case analysis that covers the entire space. Perturbation resilience assumes that the optimum clustering stands out among any alternative clustering and measures the degree by which it stands out via α. Smooth analysis is motivated by considering a problem after applying a random perturbation, which for example accounts for measurement errors.

Perturbation resilience is unique among the considered stability conditions in that we aim to recover the optimum solution, as opposed to finding a good $(1+\varepsilon)$ approximation. Awasthi et al. [35] showed that 3-perturbation resilience is sufficient to find the optimum k-median clustering, which was further improved by Balcan and Liang to $1 + \sqrt{2}$ [50] [1] and finally to 2 by Makarychev and Makarychev [238]. This is also optimal, as Balcan et al. [49] showed that recovering the optimum for $2 - \varepsilon$-perturbation resilient instances is NP-hard, see also an earlier, slightly weaker result by Ben-David and Reyzin [59] Balcan et al. [49] further showed that even for asymmetric k-center, 2-perturbation resilience is sufficient to recover the optimum.

7.1.2 Target-Based Stability

The notion of finding a target clustering is more prevalent in machine learning than minimizing an objective function. Though optimizing an objective value plays an important part in this line of research, our ultimate goal is to find a clustering C that is close to the target clustering C^*. The distance between two clusterings is the fraction of points where C and C^* disagree when considering an optimal matching of clusters in C to clusters in C^*.

When the points are generated from some (unknown) mixture model, we are also given an implicit target clustering. As a result, much work has focused on finding such clusterings using probabilistic assumptions, see, for instance, [2, 24, 57, 77, 116, 121, 122, 123, 206, 248, 281].

[1] These results also holds for a slightly more general condition called the center proximity condition.

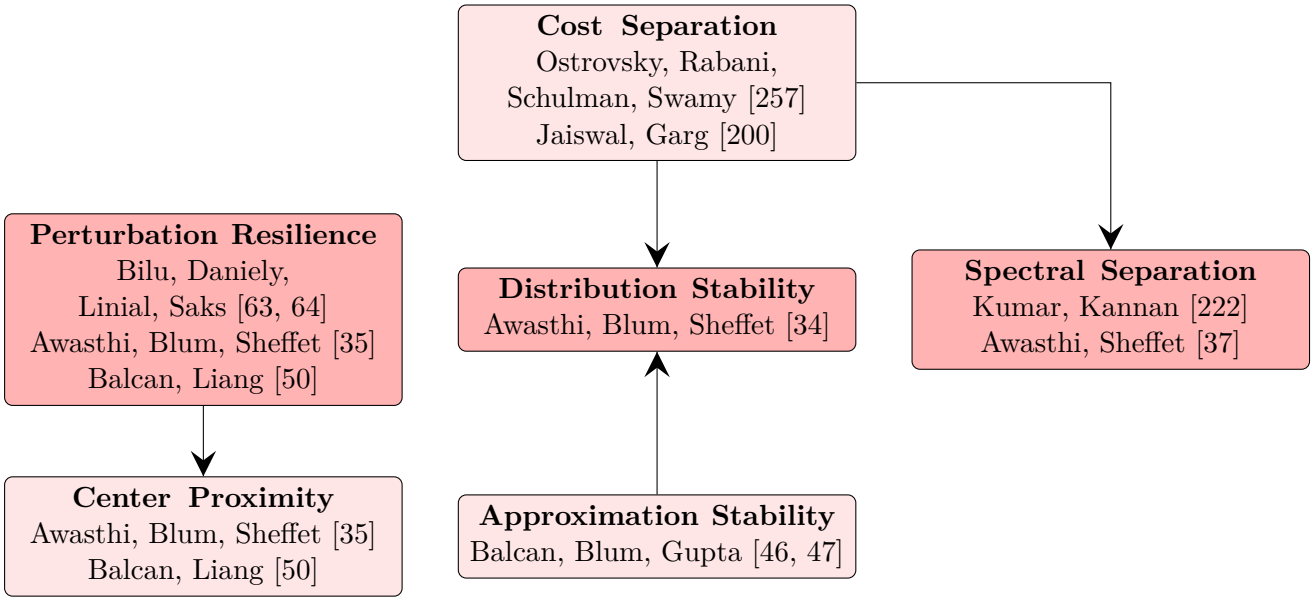

Figure 7.1: An overview over all definitions of well-clusterability. Arrows correspond to implication. For example, if an instance is cost-separated then it is distribution-stable; therefore the algorithm by Awasthi, Blum and Sheffet [34] also works for cost-separated instances. The three highlighted stability definitions in the middle of the figure are considered in this thesis.

We would like to highlight two conditions that make no probabilistic assumptions and have a particular emphasis on the k-means and k-median objective functions.

Approximation Stability The first assumption is that finding the target clustering is related to optimizing the k-means objective function. In the simplest case, the target clustering coincides with the optimum k-means clustering, but this a strong assumption that Balcan et al. [46, 47] avoid. Instead they consider instances where any clustering with cost within a factor c of the optimum has a distance at most ε to the target clustering, a condition they call (c, ε)-*approximation stability*. Balcan et al. [46, 47] then showed that this condition is sufficient to both bypass worst-case lower bounds for the approximation factor, and to find a clustering with distance $O(\varepsilon)$ from the target clustering. The condition was extended to account for the presence of noisy data by Balcan et al. [51]. This approach was improved for other min-sum clustering objectives such as correlation clustering by Balcan and Braverman [48]. For constant c, (c, ε) approximation stability also implies the β-stability condition of Awasthi et al. [34] with constant β, if the target clusters are greater than εn.

Spectral Separability Another condition that relates target clustering recovery via the k-means objective was introduced by Kumar and Kannan [222]. In order to give an intuitive

explanation, consider a mixture model consisting of k centers. If the mixture is in a low-dimensional space, and assuming that we have, for instance, approximation stability with respect to the k-means objective, we could simply use the algorithm by Balcan et al. [47]. If the mixture has many additional dimensions, the previous conditions have scaling issues, as the k-means cost may increase with each dimension, even if many of the additional dimensions mostly contain noise. The notion behind the *spectral separability* condition is that if the means of the mixture are well-separated in the subspace containing their centers, it should be possible to determine the mixture even with the added noise.

Slightly more formally, Kumar and Kannan state that a point satisfies a proximity condition if the projection of a point onto the line connecting its cluster center to another cluster center is $\Omega(k)$ standard deviations closer to its own center than to the other. The standard deviations are scaled with respect to the spectral norm of the matrix in which the ith row is the difference vector between the ith point and its cluster mean. Given that all but an ε-fraction of points satisfy the proximity condition, Kumar and Kannan [222] gave an algorithm that computes a clustering with distance $O(\varepsilon)$ to the target. They also show that their condition is (much) weaker than the cost-stability condition by Ostrovsky et al. [257] and discuss some implications of cost-stability on approximation factors. Awasthi and Sheffet [37] later showed that $\Omega(\sqrt{k})$ standard deviations are sufficient to recover most of the results by Kumar and Kannan.

7.1.3 Local Search

Local Search is an all-purpose heuristic that may be applied to any problem, see Aarts and Lenstra [1] for a general introduction. For clustering, there exists a large body of bicriteria approximations for k-median and k-means [52, 92, 112, 221]. Arya et al. [30] showed that Local Search with a neighborhood size of $1/\varepsilon$ gives a $3 + 2\varepsilon$ approximation to k-median, see also [177]. Kanungo et al. [208] proved an approximation ratio of $9 + \varepsilon$ for k-means clustering by Local Search, which was until very recently [7] the best known algorithm with a polynomial running time in metric and Euclidean spaces.[2] Recently, Local Search with an appropriate neighborhood size was shown to be a PTAS for k-means and k-median in certain restricted metrics including constant dimensional Euclidean space [111, 154]. Due to its simplicity, Local Search is also a popular subroutine for clustering tasks in various more specialized computational models [56, 65, 173]. For more theoretical clustering papers using Local Search, we refer to [115, 129, 155, 182].

Local Search is also often used for clustering in more applied areas of computer science (e. g., [278, 161, 23, 182]). Indeed, the use of Local Search with a neighborhood of size 1 for clustering was first proposed by Tüzün and Burke [278], see also Ghosh [161] for a

[2] They combined Local Search with techniques from Matousek [240] for k-means clustering in Euclidean spaces. The running time of the algorithm as stated incurs an additional factor of ε^{-d} due to the use of Matousek's approximate centroid set. Using standard techniques (see e.g. Section 7.2.1 of this thesis), a fully polynomial running time in n, d, and k is also possible without sacrificing approximation guarantees.

more efficient version of the same approach. Due the ease by which it may be implemented, Local Search has become one of the most commonly used heuristics for clustering and facility location, see Ardjmand [23]. Nevertheless, high running times is one of the biggest drawbacks of Local Search compared to other approaches, though a number of papers have engineered it to become surprisingly competitive, see Frahling and Sohler [152], Kanungo et al. [207], and Sun [272].

7.2 Distribution Stability

We consider inputs that consist in both a set of clients A and a set of candidate centers F, together with a metric distance function $\text{dist} : A \cup F \times A \cup F \to \mathbb{R}_+$. We assume that $|F|$ is polynomial in the size of the input. For Euclidean spaces, we describe a suitable discretization of the input space.

Though our analysis of distribution stability works for (k, p) clustering if p is constant, we give a slightly simpler and more readable proof by assuming $p = 1$. By introducing a dependency in $1/\varepsilon^{O(p)}$ in the neighborhood size of the algorithm and applying the two following lemmas at different steps of the proof, we ensure that the result holds for higher values of p.

Lemma 7.2.1. *Let $p \geq 0$. For any $a, b, c \in A \cup F$, we have $cost(a, b) \leq 2^p(cost(a, c) + cost(c, b))$.*

Lemma 7.2.2. *Let $p \geq 0$ and $0 \leq \varepsilon \leq 1$. For any $a, b, c \in A \cup F$, we have*

$$cost(a, b) \leq (1 + \varepsilon)^p cost(a, c) + cost(c, b)(2/\varepsilon)^p.$$

Proof. Either $\text{dist}(c, b) \leq \varepsilon \cdot \text{dist}(a, c)$ or $\text{dist}(a, c) \leq \text{dist}(c, b)/\varepsilon$. Then

$$\begin{aligned} \text{cost}(a, b) &\leq (\text{dist}(a, c) + \text{dist}(c, b))^p \leq (1 + \varepsilon)^p \text{dist}^p(a, c) + (1 + 1/\varepsilon)^p \text{dist}^p(c, b) \\ &\leq (1 + \varepsilon)^p \text{dist}^p(a, c) + \text{dist}^p(c, b)(2/\varepsilon)^p = (1 + \varepsilon)^p \text{cost}(a, c) + \text{cost}(c, b)(2/\varepsilon)^p \end{aligned}$$

□

Note that by setting $\varepsilon = 1$, Lemma 7.2.2 recovers Lemma 7.2.1.

The distribution stability condition, as originally defined by Awasthi et al. [34] is as follows.

Definition 7.2.3 (Distribution Stability [34])**.** Let (A, F, cost, k) be an input for k-clustering and let $\{C_1^*, \ldots, C_k^*\}$ denote the optimal k-clustering of A with centers $S = \{c_1^*, \ldots c_k^*\}$. Given $\beta > 0$, the instance is *β-distribution stable* if, for any i and $\forall x \notin C_i^*$,

$$\text{cost}(x, c_i^*) \geq \beta \frac{\text{OPT}}{|C_i^*|}.$$

In this chapter, we use a slightly more general definition by not requiring that $\{C_1^*, \ldots, C_k^*\}$ with its set of centers $S^* = \{c_1^*, \ldots, c_k^*\}$ is an optimal k-clustering of A. Then we say that the instance is β-*distribution stable* with respect to the clustering $\{C_1^*, \ldots, C_k^*\}$. Consequently, we require for any i and $\forall x \notin C_i^*$,

$$\text{cost}(x, c_i^*) \geq \beta \frac{\text{cost}(C^*)}{|C_i^*|}.$$

with $\text{cost}(C^*) = \sum_{i=1}^{k} \sum_{x \in C_i^*} \text{cost}(x, c_i^*)$.

We prove that Local Search is a PTAS for β-distribution stable instances. Moreover, we show that for almost all clusters (i.e.: at least $k - O(\beta^{-1}\varepsilon^{-3})$), the algorithm recovers most of the optimal clusters (i.e.: there is a bijection between the optimal clusters and the clusters of the algorithm such that a $(1-\varepsilon)$ fraction of the points of each cluster agree).

Theorem 7.2.4. *Let $\beta > 0$ and $0 < \varepsilon < 1$. For any β-distribution stable instance, the solution output by Local Search($O(\varepsilon^{-3}\beta^{-1})$) (Algorithm 10) has cost at most $(1+\varepsilon)$cost(C^*). Moreover, let $L = \{L_1, \ldots, L_k\}$ denote the clustering output by Local Search($c\varepsilon^{-3}\beta^{-1}$). There exists a bijection $\phi : L \mapsto C^*$ such that for at least $m = k - O(\beta^{-1}\varepsilon^{-3})$ clusters $L'_1, \ldots, L'_m \subseteq L$, we have both $(1-c\varepsilon)|\phi(L'_i)| \leq |L'_i \cap \phi(L'_i)|$ and $(1-c\varepsilon)|L'_i| \leq |L'_i \cap \phi(L'_i)|$, where c is an absolute constant.*

Proof outline The first important observation is that only a few clusters have more than a $1/\varepsilon^{-3}$ fraction of the total cost of the solution. For these clusters, Local Search with an appropriate neighborhood size will find an almost optimal solution. The remaining clusters are well-separated and any locally optimal solution cannot err on too many of these clusters. The cost of the points of the remaining clusters can then be charged into the overall contribution, allowing us to bound the approximation factor, see Figure 7.2. Our proof includes a few ingredients from [34] such as the notion of *inner-ring* (we work with a slightly more general definition) and distinguishing between *cheap* and *expensive* clusters. However, our analysis is more general as it allows us to analyze not only the cost of the solution of the algorithm, but also the structure of the clusters.

Remember that $C^* = \{C_1^*, \ldots, C_k^*\}$ may be the optimal clustering, in which case $\text{cost}(C^*) = \text{OPT}$, but does not have to be, in which case $\text{cost}(C^*) > \text{OPT}$. Moreover, for any cluster C_i^* and for any client $x \in C_i^*$, denote by g_x the cost of client x in solution C^*: $g_x = \text{cost}(x, c_i^*) = \text{dist}(x, c_i^*)$ since $p = 1$. Let $\mathcal{L}$ denote the output of LocalSearch($4\varepsilon^{-3}\beta^{-1} + O(\varepsilon^{-2}\beta^{-1})$) and l_x the cost induced by client x in solution $\mathcal{L}$, namely $l_x = \min_{\ell \in \mathcal{L}} \text{cost}(x, \ell)$. We have $\text{cost}(\mathcal{L}) = \sum_{x \in A} l_x \leq 5 \cdot \text{OPT} \leq 5 \cdot \text{cost}(C^*)$ due to the worst-case approximation ratio of Local Search given by Arya et al. [30].

The following definition is a generalization of the inner-ring definition of [34].

Definition 7.2.5. For any ε_0, we define the *inner ring* of cluster i, $\text{IR}_i^{\varepsilon_0}$, as the set of $x \in A \cup F$ such that $\text{cost}(x, c_i^*) \leq \varepsilon_0 \beta \text{cost}(C^*)/|C_i^*|$.

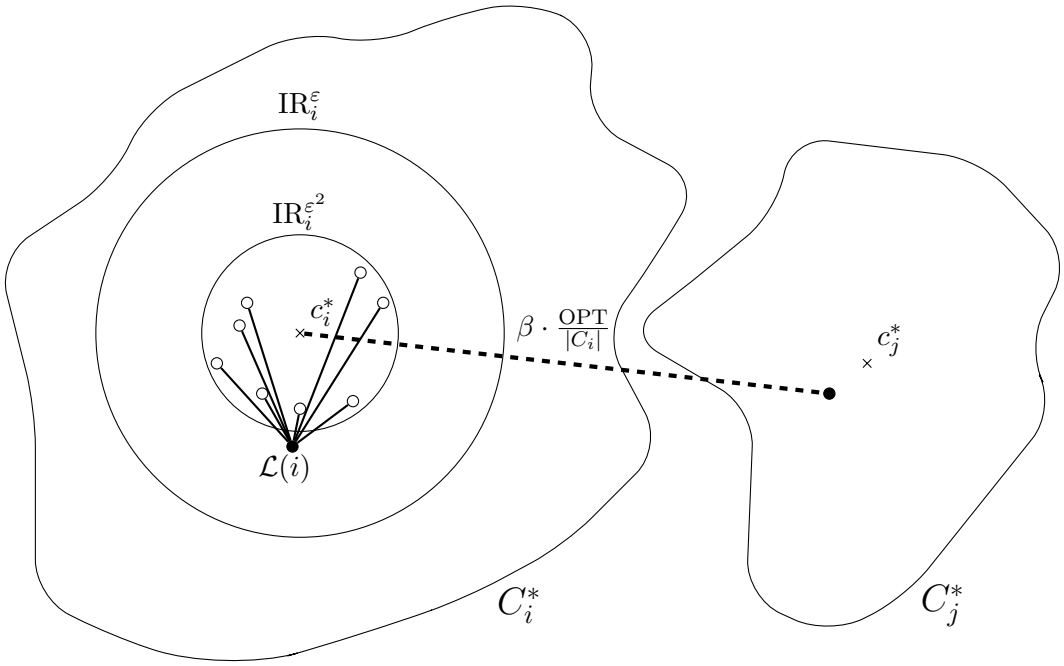

Figure 7.2: Example of a cluster $C_i^* \notin Z^*$. An important fraction of the points in $\mathrm{IR}_i^{\varepsilon^2}$ are served by $\mathcal{L}(i)$ and few points in $\bigcup_{j \neq i} C_j^*$ are served by $\mathcal{L}(i)$.

Within this chapter, we use $\varepsilon_0 \in \{\varepsilon^2, \varepsilon, 1\}$. The following lemma follows from the definition of β-distribution stability.

Lemma 7.2.6. *For any $\varepsilon_0 \leq 1$ and any cluster C_i^*, we have $IR_i^{\varepsilon_0} \subseteq C_i^*$.*

Lemma 7.2.6 directly leads to $\mathrm{IR}_i^{\varepsilon_0} \cap \mathrm{IR}_j^{\varepsilon_0} = \emptyset$ for any $\varepsilon_0 < 1$ and $i \neq j$.

We say that cluster i is *cheap* if $\sum_{x \in C_i^*} g_x \leq \varepsilon^3 \beta \mathrm{cost}(C^*)$, and *expensive* otherwise.

We also need the following lemma which generalizes Fact 4.1 in [34].

Lemma 7.2.7. *Let C_i^* be a cheap cluster. For any $\varepsilon_0 < 1$, we have $|IR_i^{\varepsilon_0}| > (1 - \varepsilon^3/\varepsilon_0)|C_i^*|$.*

Proof. Observe that each client that is not in $\mathrm{IR}_i^{\varepsilon_0}$ is at a distance larger than $\varepsilon_0 \beta \mathrm{cost}(C^*)/|C_i^*|$ from c_i^*. Since C_i^* is cheap, the total cost of the clients in $C_i^* = \mathrm{IR}_i^{\varepsilon_0} \cup (C_i^* \setminus \mathrm{IR}_i^{\varepsilon_0})$ is at most $\varepsilon^3 \beta \mathrm{cost}(C^*)$ and in particular, the total cost of the clients in $C_i^* \setminus \mathrm{IR}_i^{\varepsilon_0}$ does not exceed $\varepsilon^3 \beta \mathrm{cost}(C^*)$. Therefore, the total number of such clients is at most $\varepsilon^3 \beta \mathrm{cost}(C^*)/(\varepsilon_0 \beta \mathrm{cost}(C^*)/|C_i^*|) = \varepsilon^3 |C_i^*|/\varepsilon_0$. □

We aim at proving the following structural lemma.

Lemma 7.2.8. *There exists a set of clusters $Z^* \subseteq C^*$ of size at most $2\varepsilon^{-3}\beta^{-1} + O(\varepsilon^{-2}\beta^{-1})$ such that for any cluster $C_i^* \in C^* \setminus Z^*$, we have the following properties*

1. *C_i^* is cheap.*
2. *At least a $(1-2\varepsilon)$ fraction of C_i^* are served by a unique center $\mathcal{L}(i)$ in solution $\mathcal{L}$.*
3. *The total number of clients in $\bigcup_{j\neq i} C_j$ served by $\mathcal{L}(i)$ in $\mathcal{L}$ is at most $\varepsilon|IR_i^{\varepsilon^2}| \leq \varepsilon|C_i^*|$.*

See Fig 7.2 for a typical cluster of $C^* \setminus Z^*$. We note that parts 2 and 3 of this lemma together with Lemma 7.2.7 already imply the second part of the theorem.

For each cheap cluster C_i^*, let $\mathcal{L}(i)$ denote a center of $\mathcal{L}$ that belongs to $\mathrm{IR}_i^{\varepsilon}$ if there exists exactly one such center and remain undefined otherwise.

Lemma 7.2.9. *Let $\varepsilon < \frac{1}{3}$. Let $C^* \setminus Z_1$ denote the set of clusters C_i^* that are cheap, such that $\mathcal{L}(i)$ is defined and such that at least $(1-\varepsilon)|IR_i^{\varepsilon^2}|$ clients of $IR_i^{\varepsilon^2}$ are served in $\mathcal{L}$ by $\mathcal{L}(i)$. Then $|Z_1| \leq (2\varepsilon^{-3} + 11.25 \cdot \varepsilon^{-2} + 22.5 \cdot \varepsilon^{-1})\beta^{-1}$.*

Proof. There are five different types of clusters in C^*:

1. k_1 expensive clusters
2. k_2 cheap clusters with no center of $\mathcal{L}$ belonging to $\mathrm{IR}_i^{\varepsilon}$
3. k_3 cheap clusters with at least two centers of $\mathcal{L}$ belonging to $\mathrm{IR}_i^{\varepsilon}$
4. k_4 cheap clusters with $\mathcal{L}(i)$ being defined and less than $(1-\varepsilon)|\mathrm{IR}_i^{\varepsilon^2}|$ clients of $\mathrm{IR}_i^{\varepsilon^2}$ are served in $\mathcal{L}$ by $\mathcal{L}(i)$
5. k_5 cheap clusters with $\mathcal{L}(i)$ being defined and at least $(1-\varepsilon)|\mathrm{IR}_i^{\varepsilon^2}|$ clients of $\mathrm{IR}_i^{\varepsilon^2}$ are served in $\mathcal{L}$ by $\mathcal{L}(i)$

The definition of cheap clusters immediately yields $k_1 \leq \varepsilon^{-3}\beta^{-1}$.

Since $\mathcal{L}$ and C^* both have k clusters and the inner rings of clusters are disjoint, we have $c_1k_1 + c_3k_3 + k_4 + k_5 \leq k_1 + k_2 + k_3 + k_4 + k_5 = |Z_1| + k_5 = k$ with $c_1 \geq 0$ and $c_3 \geq 2$ resulting in $k_3 \leq (c_3 - 1)k_3 \leq (1 - c_1)k_1 + k_2 \leq k_1 + k_2$.

Before bounding k_2 and k_4, we discuss the impact of a cheap cluster C_i^* with at least a p fraction of the clients of $\mathrm{IR}_i^{\varepsilon^2}$ being served in $\mathcal{L}$ by some centers that are not in $\mathrm{IR}_i^{\varepsilon}$. By the triangular inequality, the cost in $\mathcal{L}$ for any client x of this p fraction is at least $(\varepsilon - \varepsilon^2)\beta\mathrm{cost}(C^*)/|C_i^*|$. Then the total cost of all clients of this p fraction in $\mathcal{L}$ is at least $p|\mathrm{IR}_i^{\varepsilon^2}|(1-\varepsilon)\varepsilon\beta\mathrm{cost}(C^*)/|C_i^*|$. By Lemma 7.2.7, substituting $|\mathrm{IR}_i^{\varepsilon^2}|$ with $(1-\varepsilon)|C_i^*|$ yields for this total cost

$$p|\mathrm{IR}_i^{\varepsilon^2}|(1-\varepsilon)\varepsilon\beta\frac{\mathrm{cost}(C^*)}{|C_i^*|} \geq p(1-\varepsilon)^2|C_i^*|\varepsilon\beta\frac{\mathrm{cost}(C^*)}{|C_i^*|} = p(1-\varepsilon)^2\varepsilon\beta\mathrm{cost}(C^*).$$

To determine k_2, we must use $p = 1$ while we have $p > \varepsilon$ for k_4. Therefore, the total costs of all clients of the k_2 and the k_4 clusters in $\mathcal{L}$ are at least $k_2(1-\varepsilon)^2\varepsilon\beta\text{cost}(C^*)$ and $k_4(1-\varepsilon)^2\varepsilon^2\beta\text{cost}(C^*)$, respectively.

Now, since $\text{cost}(\mathcal{L}) \leq 5\text{OPT} \leq 5\text{cost}(C^*)$, we have $(k_2 + k_4\varepsilon)\varepsilon\beta \leq 5/(1-\varepsilon)^2 \leq 45/4$.

Therefore, we have $|Z_1| = k_1 + k_2 + k_3 + k_4 \leq 2k_1 + 2k_2 + k_4 \leq (2\varepsilon^{-3} + 11.25 \cdot \varepsilon^{-2} + 22.5 \cdot \varepsilon^{-1})\beta^{-1}$. □

We continue with the following lemma whose proof relies on similar arguments.

Lemma 7.2.10. *There exists a set $Z_2 \subseteq C^* \setminus Z_1$ of size at most $11.25\varepsilon^{-1}\beta^{-1}$ such that for any cluster $C_j^* \in C^* \setminus Z_2$, the total number of clients $x \in \bigcup_{i \neq j} C_i$, that are served by $\mathcal{L}(j)$ in $\mathcal{L}$, is at most $\varepsilon|IR_i^{\varepsilon^2}|$.*

Proof. Consider a cheap cluster $C_j^* \in C^* \setminus Z_1$ such that the total number of clients $x \in C_i$ for $i \neq j$, that are served by $\mathcal{L}(j)$ in $\mathcal{L}$, is greater than $\varepsilon|\text{IR}_j^{\varepsilon^2}|$. By the triangular inequality and the definition of β-distribution stability, the total cost for each $x \in C_i$ with $i \neq j$ served by $\mathcal{L}(j) \in \mathcal{L}$ is at least $(1-\varepsilon)\beta\text{cost}(C^*)/|C_j^*|$. Since there are at least $\varepsilon|\text{IR}_j^{\varepsilon^2}|$ such clients, their total cost is at least $\varepsilon|\text{IR}_j^{\varepsilon^2}|(1-\varepsilon)\beta\text{cost}(C^*)/|C_j^*|$. By Lemma 7.2.7, this total cost is at least

$$\varepsilon|\text{IR}_j^{\varepsilon^2}|(1-\varepsilon)\beta\frac{\text{cost}(C^*)}{|C_j^*|} \geq \varepsilon(1-\varepsilon)^2|C_j^*|\beta\frac{\text{cost}(C^*)}{|C_j^*|}.$$

Recall that by [30], $\mathcal{L}$ is a 5-approximation and so there exist at most $11.25 \cdot \varepsilon^{-1}\beta^{-1}$ such clusters. □

Therefore, the proof of Lemma 7.2.8 follows from combining Lemmas 7.2.9 and 7.2.10 and by considering that $(1-\varepsilon)|\text{IR}_i^{eps^2}| \geq (1-\varepsilon)^2|C_i^*| \geq (1-2\varepsilon)|C_i^*|$ (Lemma 7.2.7) holds.

We now turn to the analysis of the cost of $\mathcal{L}$. Let $C(Z^*) = \bigcup_{C_i^* \in Z^*} C_i^*$. For any cluster $C_i^* \in C^* \setminus Z^*$, let $\mathcal{L}(i)$ be the unique center of $\mathcal{L}$ that serves at least $(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}| > (1-\varepsilon)^2|C_i|$ clients of $\text{IR}_i^{\varepsilon^2}$, see Lemmas 7.2.8 and 7.2.9. Let $\widehat{\mathcal{L}} = \bigcup_{C_i^* \in C^* \setminus Z^*} \mathcal{L}(i)$ and define $\widehat{A}$ to be the set of clients that are served in solution $\mathcal{L}$ by centers of $\widehat{\mathcal{L}}$. Finally, let $A(\mathcal{L}(i))$ be the set of clients that are served by $\mathcal{L}(i)$ in solution $\mathcal{L}$. Observe that the $A(\mathcal{L}(i))$ partition $\widehat{A}$.

Lemma 7.2.11. *We have*

$$-\varepsilon \cdot \textit{cost}(\mathcal{L})/n + \sum_{x \in (A \setminus \widehat{A}) \cup C(Z^*)} l_x \leq \sum_{x \in (A \setminus \widehat{A}) \cup C(Z^*)} g_x + \frac{2\varepsilon}{(1-\varepsilon)^2} \cdot (\textit{cost}(C^*) + \textit{cost}(\mathcal{L})).$$

Proof. Consider the following mixed solution $\mathcal{M} = \widehat{\mathcal{L}} \cup \{c_i^* \mid C_i^* \in Z^*\}$. We start by bounding the cost of $\mathcal{M}$. For any client $x \in \widehat{A}$, the center that serves it in $\mathcal{L}$ belongs to $\mathcal{M}$. Thus its cost in $\mathcal{M}$ is at most l_x. Now, for any client $x \in C(Z^*)$, the center that serves it in C^* is in $\mathcal{M}$, so its cost in $\mathcal{M}$ is at most g_x.

Finally, we evaluate the cost of the clients in $A \setminus (\widehat{A} \cup C(Z^*))$. Consider such a client x and let C_i^* be the cluster it belongs to in solution C^*. Since $C_i^* \in C^* \setminus Z^*$, $\mathcal{L}(i)$ is defined and we have $\mathcal{L}(i) \in \widehat{\mathcal{L}} \subseteq \mathcal{M}$. Hence, the cost of x in $\mathcal{M}$ is at most $\text{cost}(x, \mathcal{L}(i))$. Observe that by the triangular inequality, $\text{cost}(x, \mathcal{L}(i)) \leq \text{cost}(x, c_i^*) + \text{cost}(c_i^*, \mathcal{L}(i)) = g_x + \text{cost}(c_i^*, \mathcal{L}(i)) \leq g_x + \varepsilon\beta\text{cost}(C^*)/|C_i^*|$.

Now consider a client $x' \in \text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))$. By the triangular inequality, we have $\text{cost}(c_i^*, \mathcal{L}(i)) \leq \text{cost}(c_i^*, x') + \text{cost}(x', \mathcal{L}(i)) = g_{x'} + l_{x'}$. Hence,

$$\text{cost}(c_i^*, \mathcal{L}(i)) \leq \frac{1}{|\text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))|} \sum_{x' \in \text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))} (g_{x'} + l_{x'}).$$

It follows that assigning the clients of $C_i^* \cap (A \setminus \widehat{A})$ to $\mathcal{L}(i)$ induces a cost of at most

$$\sum_{x \in C_i^* \cap (A \setminus \widehat{A})} g_x + \frac{|C_i^* \cap (A \setminus \widehat{A})|}{|\text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))|} \sum_{x' \in \text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))} (g_{x'} + l_{x'}).$$

Due to Lemma 7.2.9, we have $|\text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))| \geq (1-\varepsilon) \cdot |\text{IR}_i^{\varepsilon^2}|$ and $|\text{IR}_i^{\varepsilon^2} \cap (A \setminus \widehat{A})| \leq \varepsilon \cdot |\text{IR}_i^{\varepsilon^2}|$. Further, $|(C_i^* \setminus \text{IR}_i^{\varepsilon^2}) \cap (A \setminus \widehat{A})| \leq |(C_i^* \setminus \text{IR}_i^{\varepsilon^2})| = |C_i^*| - |\text{IR}_i^{\varepsilon^2}|$. Combining these three bounds, we have

$$\begin{aligned}
\frac{|C_i^* \cap (A \setminus \widehat{A})|}{|\text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))|} &= \frac{|(C_i^* \setminus \text{IR}_i^{\varepsilon^2}) \cap (A \setminus \widehat{A})| + |\text{IR}_i^{\varepsilon^2} \cap (A \setminus \widehat{A})|}{|\text{IR}_i^{\varepsilon^2} \cap A(\mathcal{L}(i))|} \\
&\leq \frac{|C_i^*| - (1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|}{(1-\varepsilon) \cdot |\text{IR}_i^{\varepsilon^2}|} = \frac{|C_i^*|}{(1-\varepsilon) \cdot |\text{IR}_i^{\varepsilon^2}|} - 1 \\
&\leq \frac{|C_i^*|}{(1-\varepsilon)^2 \cdot |C_i^*|} - 1 \leq \frac{2\varepsilon - \varepsilon^2}{(1-\varepsilon)^2} < \frac{2\varepsilon}{(1-\varepsilon)^2}, \qquad (7.1)
\end{aligned}$$

where the inequality in 7.1 follows from Lemma 7.2.7.

Summing over all clusters $C_i^* \in C^* \setminus Z^*$, we obtain that the cost in $\mathcal{M}$ for the clients in $A \setminus (\widehat{A} \cup C(Z^*))$ is less than

$$\sum_{x \in A \setminus (\widehat{A} \cup C(Z^*))} g_x + \frac{2\varepsilon}{(1-\varepsilon)^2} \cdot (\text{cost}(C^*) + \text{cost}(\mathcal{L})).$$

By Lemmas 7.2.9 and 7.2.10, we have $|\mathcal{M} \setminus \mathcal{L}| + |\mathcal{L} \setminus \mathcal{M}| = 2 \cdot |Z^*| \leq (4\varepsilon^{-3} + O(\varepsilon^{-2}))\beta^{-1}$. By selecting the neighborhood size of Local Search (Algorithm 10) to be greater than this value, we have $(1 - \varepsilon/n) \cdot \text{cost}(\mathcal{L}) \leq \text{cost}(\mathcal{M})$. Therefore, combining the above observations, we have

$$(1 - \frac{\varepsilon}{n}) \cdot \text{cost}(\mathcal{L}) < \sum_{x \in \widehat{A} \setminus C(Z^*)} l_x + \sum_{x \in C(Z^*)} g_x + \sum_{x \in A \setminus (\widehat{A} \cup C(Z^*))} g_x + \frac{2\varepsilon}{(1-\varepsilon)^2} \cdot (\text{cost}(C^*) + \text{cost}(\mathcal{L})).$$

By simple transformations, we then obtain

$$-\frac{\varepsilon}{n}\cdot \text{cost}(\mathcal{L}) + \sum_{x\in(A\setminus\widehat{A})\cup C(Z^*)} l_x < \sum_{x\in(A\setminus\widehat{A})\cup C(Z^*)} g_x + \frac{2\varepsilon}{(1-\varepsilon)^2}\cdot(\text{cost}(C^*)+\text{cost}(\mathcal{L})).$$

□

We now turn to evaluate the cost for the clients that are in $\widehat{A}\setminus C(Z^*)$. For any cluster $C_i^*\in C^*\setminus C(Z^*)$ and for any $x\in C_i^*\setminus A(\mathcal{L}(i))$ define Reassign(x) to be the cost of x with respect to the center in $\mathcal{L}(i)$. Note that there exists only one center of $\mathcal{L}$ in IR_i^ε for any cluster $C_i^*\in C^*\setminus C(Z^*)$. Before going deeper in the analysis, we need the following lemma.

Lemma 7.2.12. *For any $C_i^*\in C^*\setminus C(Z^*)$, we have*

$$\sum_{x\in C_i^*\setminus A(\mathcal{L}(i))} \textit{Reassign}(x) \le \sum_{x\in C_i^*\setminus A(\mathcal{L}(i))} g_x + \frac{2\varepsilon}{(1-\varepsilon)^2}\sum_{x\in C_i^*\cap A(\mathcal{L}_i)}(l_x+g_x).$$

Proof. Consider a client $x\in C_i^*\setminus A(\mathcal{L}(i))$. By the triangular inequality, we have Reassign(x) = $\text{cost}(x,\mathcal{L}(i))\le \text{cost}(x,c_i^*)+\text{cost}(c_i^*,\mathcal{L}(i)) = g_x+\text{cost}(c_i^*,\mathcal{L}(i))$. Then,

$$\sum_{x\in C_i^*\setminus A(\mathcal{L}(i))}\text{Reassign}(x)\le \sum_{x\in C_i^*\setminus A(\mathcal{L}(i))} g_x + |C_i^*\setminus A(\mathcal{L}(i))|\cdot\text{cost}(c_i^*,\mathcal{L}(i)).$$

Now consider a client $x'\in C_i^*\cap A(\mathcal{L}(i))$. By the triangular inequality, we have $\text{cost}(c_i^*,\mathcal{L}(i))\le \text{cost}(c_i^*,x')+\text{cost}(x',\mathcal{L}(i))\le g_x+l_x$. Therefore,

$$\text{cost}(c_i^*,\mathcal{L}(i))\le \frac{1}{|C_i^*\cap A(\mathcal{L}(i))|}\sum_{x\in C_i^*\cap A(\mathcal{L}(i))}(g_x+l_x).$$

We now bound $\frac{|C_i^*\setminus A(\mathcal{L}(i))|}{|C_i^*\cap A(\mathcal{L}(i))|}$. Due to Lemma 7.2.7, we have $|\text{IR}_i^{\varepsilon^2}|\ge(1-\varepsilon)|C_i^*|$ and due to Lemma 7.2.8, we have $|\text{IR}_i^{\varepsilon^2}\cap A(\mathcal{L}(i))|\ge(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|$. Therefore $|C_i^*\cap A(\mathcal{L}(i))|\ge(1-\varepsilon)^2|C_i^*|$ and $|C_i^*\setminus A(\mathcal{L}(i))|\le(1-(1-\varepsilon)^2)|C_i^*|\le 2\varepsilon|C_i^*|$, yielding $\frac{|C_i^*\setminus A(\mathcal{L}(i))|}{|C_i^*\cap A(\mathcal{L}(i))|}\le\frac{2\varepsilon}{(1-\varepsilon)^2}$.

Combining, we obtain

$$\begin{aligned}\sum_{x\in C_i^*\setminus A(\mathcal{L}(i))}\text{Reassign}(x) &\le \sum_{x\in C_i^*\setminus A(\mathcal{L}(i))} g_x + \frac{|C_i^*\setminus A(\mathcal{L}(i))|}{|C_i^*\cap A(\mathcal{L}(i))|}\sum_{x\in C_i^*\cap A(\mathcal{L}(i))}(g_x+l_x)\\ &\le \sum_{x\in C_i^*\setminus A(\mathcal{L}(i))} g_x + \frac{2\varepsilon}{(1-\varepsilon)^2}\sum_{x\in C_i^*\cap A(\mathcal{L}(i))}(g_x+l_x).\end{aligned}$$

□

We now partition the clients of cluster $C_i^*\in C^*\setminus Z^*$. We use the following definitions: $B_{i,j}$ is the set of clients of C_i^* that are served in solution $\mathcal{L}$ by center $\mathcal{L}(j)$ for $i\neq j$. Then

we set $B_i = \bigcup_j B_{i,j}$ and $D_j = \bigcup_i B_{i,j}$. We further define $E_i = \left(C_i^* \cap \widehat{A}\right) \setminus \bigcup_{j \neq i} D_j = \left(C_i^* \cap \widehat{A}\right) \setminus \left(\bigcup_{j \neq i} \bigcup_i B_{i,j}\right) = C_i^* \cap A(\mathcal{L}(i))$.

Lemma 7.2.13. *Let C_i^* be a cluster in $C^* \setminus Z^*$. Define the solution $\mathcal{M}^i = \mathcal{L} \setminus \{\mathcal{L}(i)\} \cup \{c_i^*\}$ and denote by m_x^i the cost of client x in solution $\mathcal{M}^i$. Then*

$$\sum_{x \in A} m_x^i \leq \sum_{x \in A \setminus A(\mathcal{L}(i))} l_x + \sum_{x \in E_i} g_x + \sum_{x \in D_i} \mathit{Reassign}(x) + \sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} l_x + \frac{\varepsilon}{(1-\varepsilon)} \sum_{x \in E_i} (g_x + l_x).$$

Proof. For any client $x \in A \setminus A(\mathcal{L}(i))$, the center that serves it in $\mathcal{L}$ belongs to $\mathcal{M}^i$. Thus its cost is at most l_x. Moreover, observe that any client $x \in E_i \subseteq C_i^*$ can now be served by c_i^*, and so its cost is at most g_x. For each client $x \in D_i$, we bound its cost by Reassign(x) since all the centers of $\mathcal{L}$ except for $\mathcal{L}(i)$ are in $\mathcal{M}^i$.

Now, we bound the cost of a client $x \in A(\mathcal{L}(i)) \setminus (E_i \cup D_i) \subseteq A(\mathcal{L}(i)) \setminus C_i^*$. The closest center in $\mathcal{M}^i$ for a client $x' \in A(\mathcal{L}(i))$ is not farther than c_i^*. By the triangular inequality, the cost of such client x' is at most $\text{cost}(x', c_i^*) \leq \text{cost}(x', \mathcal{L}(i)) + \text{cost}(\mathcal{L}(i), c_i^*) = l_{x'} + \text{cost}(\mathcal{L}(i), c_i^*)$, and so

$$\sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} m_x^i \leq |A(\mathcal{L}(i)) \setminus C_i^*| \cdot \text{cost}(\mathcal{L}(i), c_i^*) + \sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} l_x. \tag{7.2}$$

For any client $x \in E_i = |A(\mathcal{L}(i)) \cap C_i^*|$, the triangular inequality yields $\text{cost}(\mathcal{L}(i), c_i^*) \leq \text{cost}(\mathcal{L}(i), x) + \text{cost}(x, c_i^*) = l_x + g_x$. Therefore,

$$\text{cost}(\mathcal{L}(i), c_i^*) \leq \frac{1}{|A(\mathcal{L}(i)) \cap C_i^*|} \sum_{x \in E_i} (l_x + g_x). \tag{7.3}$$

Combining Equations 7.2 and 7.3, we have

$$\sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} m_x^i \leq \sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} l_x + \frac{|A(\mathcal{L}(i)) \setminus C_i^*|}{|A(\mathcal{L}(i)) \cap C_i^*|} \sum_{x \in E_i} (l_x + g_x) \tag{7.4}$$

We now remark that since C_i^* is in $C^* \setminus Z^*$, we have by Lemmas 7.2.8 and 7.2.7, $|A(\mathcal{L}(i)) \setminus C_i^*| \leq |A(\mathcal{L}(i)) \setminus \text{IR}_i^{\varepsilon^2}| \leq \varepsilon \cdot |\text{IR}_i^{\varepsilon^2}| \leq \varepsilon(1-\varepsilon) \cdot |C_i^*|$ and $(1-\varepsilon)^2 \cdot |C_i^*| \leq (1-\varepsilon) \cdot |\text{IR}_i^{\varepsilon^2}| \leq |A(\mathcal{L}(i)) \cap \text{IR}_i^{\varepsilon^2}| \leq |A(\mathcal{L}(i)) \cap C_i|$. Thus, combining with Equation 7.4 yields

$$\sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} m_x^i \leq \sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} l_x + \frac{\varepsilon}{1-\varepsilon} \sum_{x \in E_i} (l_x + g_x)$$

which concludes the proof. □

We can thus prove the final lemma of this section.

Lemma 7.2.14. *We have*

$$-\varepsilon \cdot \mathit{cost}(\mathcal{L}) + \sum_{x \in \widehat{A} \setminus C(Z^*)} l_x \leq \sum_{x \in \widehat{A} \setminus C(Z^*)} g_x + \frac{3\varepsilon}{(1-\varepsilon)^2} \cdot (\mathit{cost}(\mathcal{L}) + \mathit{cost}(C^*)).$$

Proof. We consider a cluster C_i^* in $C^* \setminus Z^*$ and the solution $\mathcal{M}^i = \mathcal{L} \setminus \{\mathcal{L}(i)\} \cup \{c_i^*\}$. Observe that $\mathcal{M}^i$ and $\mathcal{L}$ only differ by $\mathcal{L}(i)$ and c_i^*. Therefore, by local optimality we have $(1 - \frac{\varepsilon}{n}) \cdot \mathrm{cost}(\mathcal{L}_i) \leq \mathrm{cost}(\mathcal{M}^i)$. Then Lemma 7.2.13 yields

$$(1-\frac{\varepsilon}{n})\cdot\mathrm{cost}(\mathcal{L}_i) \leq \sum_{x \in A \setminus A(\mathcal{L}_i))} l_x + \sum_{x \in E_i} g_x + \sum_{x \in D_i} \mathrm{Reassign}(x) + \sum_{\substack{x \in A(\mathcal{L}(i)) \setminus \\ (E_i \cup D_i)}} l_x + \frac{\varepsilon}{(1-\varepsilon)} \cdot \sum_{x \in E_i} (g_x + l_x)$$

and so, simplifying

$$-\frac{\varepsilon}{n} \cdot \mathrm{cost}(\mathcal{L}_i) + \sum_{x \in E_i} l_x + \sum_{x \in D_i} l_x \leq \sum_{x \in E_i} g_x + \sum_{x \in D_i} \mathrm{Reassign}(x) + \frac{\varepsilon}{(1-\varepsilon)} \cdot \sum_{x \in E_i} (g_x + l_x)$$

We now apply this analysis to each cluster $C_i^* \in C^* \setminus Z^*$. Summing over all clusters C_i^*, we obtain,

$$-\frac{\varepsilon}{n} \cdot \mathrm{cost}(\mathcal{L}) + \sum_{i=1}^{|C^* \setminus Z^*|} \left(\sum_{x \in E_i} l_x + \sum_{x \in D_i} l_x \right) \leq \sum_{i=1}^{|C^* \setminus Z^*|} \left(\sum_{x \in E_i} g_x + \sum_{x \in D_i} \mathrm{Reassign}(c) \right) + \frac{\varepsilon}{(1-\varepsilon)} \cdot (\mathrm{cost}(\mathcal{L}) + \mathrm{cost}(C^*))$$

By Lemma 7.2.12, we have

$$-\frac{\varepsilon}{n} \cdot \mathrm{cost}(\mathcal{L}) + \sum_{i=1}^{|C^* \setminus Z^*|} \sum_{x \in C_i^* \cap \widehat{A}} l_x \leq \sum_{i=1}^{|C^* \setminus Z^*|} \sum_{x \in C_i^* \cap \widehat{A}} g_x + \left(\frac{\varepsilon}{1-\varepsilon} + \frac{2\varepsilon}{(1-\varepsilon)^2} \right) \cdot (\mathrm{cost}(\mathcal{L}) + \mathrm{cost}(C^*)).$$

Therefore, $-\frac{\varepsilon}{n} \cdot \mathrm{cost}(\mathcal{L}) + \sum_{x \in \widehat{A} \setminus C(Z^*)} l_x \leq \sum_{x \in \widehat{A} \setminus C(Z^*)} g_x + \frac{3\varepsilon}{(1-\varepsilon)^2} \cdot (\mathrm{cost}(\mathcal{L}) + \mathrm{cost}(C^*))$. □

Finally, we prove Theorem 7.2.4.

Proof of Theorem 7.2.4. Lemma 7.2.8 directly implies the second part of the theorem. For

the first part, we sum the equations from Lemmas 7.2.11 and 7.2.14 and obtain

$$
\begin{aligned}
&-\frac{\varepsilon}{n}\cdot\text{cost}(\mathcal{L})+\sum_{x\in(A\setminus\widehat{A})\cup C(Z^*)} l_x-\frac{\varepsilon}{n}\cdot\text{cost}(\mathcal{L})+\sum_{x\in\widehat{A}\setminus C(Z^*)} l_x\\
\leq&\sum_{x\in(A\setminus\widehat{A})\cup C(Z^*)} g_x+\frac{2\varepsilon}{(1-\varepsilon)^2}(\text{cost}(C^*)+\text{cost}(\mathcal{L}))+\\
&\sum_{x\in\widehat{A}\setminus C(Z^*)} g_x+\frac{3\varepsilon}{(1-\varepsilon)^2}(\text{cost}(\mathcal{L})+\text{cost}(C^*))\\
\Rightarrow&\text{cost}(\mathcal{L})\leq\text{cost}(C^*)+\left(\varepsilon\cdot\left(\frac{5}{(1-\varepsilon)^2}+\frac{2}{n}\right)\right)(\text{cost}(\mathcal{L})+\text{cost}(C^*)).
\end{aligned}
$$

□

7.2.1 Euclidean Distribution Stability

In this section we show how to reduce the Euclidean problem to the discrete version. For constant p, we obtain polynomial sized candidate solution sets in polynomial time. For k-means itself, we could alternatively combine Matousek's approximate centroid set [240] with the Johnson Lindenstrauss lemma and avoid the following construction. Roughly, our approach is to (1) reduce the target dimension such that it is only logarithmic w.r.t the size of the input and (2) provide a sufficient fine ε-net, though there are a few technicalities. We emphasize that the techniques are fairly standard in this line of research and that the analysis is similar to coreset papers by Har-Peled and Mazumdar [185], Frahling and Sohler [151], and Fichtenberger et al. [149].

First, we describe a discretization procedure. It will be important to us that the candidate solution preserves (1) the cost of any given set of centers and (2) distribution stability. We first recall the definition of ε-nets for Euclidean spaces.

Definition 7.2.15. Let S be the unit sphere in $\mathbb{R}^d$ and $\varepsilon > 0$. A set of points $\mathcal{N}_\varepsilon$ is an *ε-net* of S if for every point $x \in S$ there exists some point $y \in \mathcal{N}_\varepsilon$ with $||x - y|| \leq \varepsilon$.

It is well known that for unit Euclidean ball of dimension d, there exists an ε-net of cardinality $(1 + 2/\varepsilon)^d$, see for instance Pisier [262], though in this case the proof is non-constructive. Constructive methods yield slightly worse, but asymptotically similar bounds of the form $\varepsilon^{-O(d)}$, see for instance Chazelle [94] for an extensive overview on how to construct such nets. Note that having constructed an ε-net for the unit sphere, we also have an $\varepsilon \cdot r$-net for any sphere with radius r. The following lemma shows that a sufficiently small ε-net preserves distribution stability. Again for ease of exposition, we only give the proof for k-median ($(k, 1)$ clustering), and assuming we can construct an appropriate ε-net, but similar results also hold for (k, p) clustering as long as p is constant.

Lemma 7.2.16. *Let A be a set of n points in d-dimensional Euclidean space and let $\beta, \varepsilon > 0$ with $\min(1, \beta, \varepsilon) > 2\eta > 0$ be constants. Suppose there exists a clustering $C = \{C_1, \ldots, C_k\}$ with centers $S = \{c_1, \ldots c_k\}$ such that*

1. *$cost(C, S) = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - c_i||$ is a constant approximation to the optimum clustering and*
2. *C is β-distribution stable.*

Then there exists a discretization D of the solution space such that there exists a subset $S' = \{c'_1, \ldots c'_k\} \subset D$ of size k with

1. *$\sum_{i=1}^{k} \sum_{x \in C_i} ||x - c'_i|| \leq (1 + \varepsilon) \cdot cost(C, S)$ and*
2. *C with centers S' is $\beta/2$-distribution stable.*

The discretization consists of $O(n \cdot \log n \cdot \eta^{-(d+2)})$ many points.

Proof. Let OPT being the cost of an optimal k-median clustering. Define an exponential sequence to the base of $(1 + \eta)$ starting at $(\eta \cdot \frac{\mathrm{OPT}}{n})$ and ending at $(n \cdot \mathrm{OPT})$. The sequence contains $t = \log_{1+\eta}(n^2/\eta) + 1$ many elements. For each point $x \in A$, define $B(x, r_i)$ as the d-dimensional ball centered at x with radius $r_i = (1 + \eta)^i \cdot \eta \cdot \frac{\mathrm{OPT}}{n}$. We cover the ball $B(x, r_i)$ with an $\eta/8 \cdot r_i$ net denoted by $\mathcal{N}_{\eta/8}(x, r_i)$. As the set of candidate centers, we set $D = \cup_{x \in A} \cup_{i=0}^{t} \mathcal{N}_{\eta/8}(x, r_i)$. Clearly, $|D| = n \cdot (t + 1) \cdot (1 + 16/\eta)^d$. Due to the Taylor expansion of $\ln(1 + \eta)$ and $-\log \eta < \eta^{-1}$ and assuming d is a constant, we have $|D| \in O(n \cdot \log n \cdot (1 + 16/\eta)^{d+2}) = O(n \log n \cdot \eta^{-(d+2)})$.

Now for each $c_j \in S$, set $c'_j = \operatorname{argmin}_{q \in D} ||q - c_j||$. We will show that $S' = \{c'_1, \ldots c'_k\}$ satisfies the two conditions of the lemma.

For (1), we first consider the points $x \in C_j$ with $||x - c_j|| \leq \eta \cdot \frac{\mathrm{OPT}}{n}$. Then there exists a $c'_j \in D$ such that $||x - c'_j|| \leq (9\eta/8)\frac{\mathrm{OPT}}{n}$.

Now consider the remaining points of C_j. Since the $\mathrm{cost}(C, S)$ is a constant approximation, we have $||x - c_j|| \leq n \cdot \mathrm{OPT}$. Therefore for these points x, the center c_j of x satisfies $(1+\eta)^i \cdot \eta \cdot \frac{\mathrm{OPT}}{n} \leq ||x - c_j|| \leq (1+\eta)^{i+1} \cdot \eta \cdot \frac{\mathrm{OPT}}{n}$ for some $i \in \{0, \ldots t\}$. Then there exists some point $c'_j \in \mathcal{N}_{\eta/8}(x, r_{i+1})$ with $||c'_j - c_j|| \leq \eta/8 \cdot (1+\eta)^{i+1} \cdot \eta \cdot \frac{\mathrm{OPT}}{n} \leq \eta/8 \cdot (1+\eta)||x - c_i||$. We then have $||x - c'_j|| \leq (1 + \eta/8 \cdot (1+\eta))||x - c_j||$. Summing up over all points, we have a total cost of at most $9\eta/8 \cdot \mathrm{OPT} + (1 + \eta/8 \cdot (1+\eta)) \cdot \mathrm{cost}(C, S') \leq (1 + \frac{11\eta}{8}) \cdot \mathrm{cost}(C, S') \leq (1+\varepsilon) \cdot \mathrm{cost}(C, S')$.

To show (2), let us consider some point $y \notin C_j$. If $\beta \cdot \frac{\mathrm{cost}(C,S)}{|C_j|} > \eta \cdot \frac{\mathrm{OPT}}{n}$, then $||y - c'_j|| \geq ||y - c_j|| - ||c'_j - c_j|| > (1 - \eta/8 \cdot (1 + \eta)) \cdot ||y - c_j|| > \frac{3}{4}\beta \cdot \frac{\mathrm{cost}(C,S)}{|C_j|}$. Otherwise, we know that due to $2\eta < \beta$ and the fact that we cover $B(y, r_0)$ with an $\mathcal{N}_{\eta/8}(y, r_0)$ net, we have $||c_j - c'_j|| \leq \eta^2/8 \cdot \frac{\mathrm{OPT}}{n} \leq \frac{\beta}{16} \cdot \frac{\mathrm{cost}(C,S)}{|C_j|}$. By the triangle inequality $||y - c'_j|| \geq ||y - c_j|| - ||c_j - c'_j|| \geq \beta \cdot \frac{\mathrm{cost}(C,S)}{|C_j|} \cdot (1 - \frac{1}{16})$. □

To reduce the dependency on the dimension, we combine this statement with standard dimension reduction techniques.

Lemma 7.2.17 (Johnson-Lindenstrauss Lemma [202])**.** *For any set of n points A in d-dimensional Euclidean space and any $0 < \varepsilon < 1/2$, there exists a distribution $\mathcal{F}$ over linear maps $f : \ell_2^d \to \ell_2^m$ with $m \in O(\varepsilon^{-2} \log n)$ such that for all point pairs (x, y) in A with probability at least $2/3$*

$$(1 - \varepsilon) \cdot ||x - y|| \leq ||f(x) - f(y)|| \leq (1 + \varepsilon) \cdot ||x - y||.$$

For a given clustering C that has β distribution stability, the Johnson-Lindenstrauss lemma preserves both cost (up to a $(1+\varepsilon)$ factor) and stability (up to a $(1-\varepsilon)$ factor) by setting the target dimension $m \in O(\varepsilon^{-2} \log(n \cdot k)) = O(\varepsilon^{-2} \log n^2) = O(\varepsilon^{-2} \log n)$. Increasing the target dimension slightly allows us to preserve the cost of all clusterings.

Lemma 7.2.18 (Cost-preserving Projections [216])**.** *For any set of n points A in d-dimensional Euclidean space and any $0 < \varepsilon < 1/2$, there exists a distribution $\mathcal{F}$ over linear maps $f : \ell_2^d \to \ell_2^m$ with $m \in O(\varepsilon^{-3} \log(1/\varepsilon) \log n)$ such that for all sets of k points S with probability at least $2/3$*

$$(1 - \varepsilon) \sum_{x \in A} \min_{c \in S} dist^p(x, c) \leq \sum_{x \in A} \min_{c \in S} dist^p(f(x), f(c)) \leq (1 + \varepsilon) \sum_{x \in A} \min_{c \in S} dist^p(x, c).$$

Combining Lemmas 7.2.16, 7.2.17, and 7.2.18 gives us the following corollary.

Corollary 7.2.19. *Let A be a set of points in d-dimensional Euclidean space with a clustering $C = \{C_1, \ldots C_k\}$ and centers $S = \{c_1, \ldots c_k\}$ such that C is β-distribution stable. Then there exists a (k, p)-clustering instance with clients A, $n^{poly(\varepsilon^{-1})}$ centers F and a subset $S' \subset F \cup A$ of k centers such that C and S' is $\Omega(\beta)$ stable and the cost of clustering A with S' is at most $(1 + \varepsilon)$ times the cost of clustering A with S.*

7.3 Perturbation Resilience

We consider the definition of α-perturbation-resilient clustering given by Awasthi et al. [35].

Definition 7.3.1 (Perturbation Resilience [35])**.** Let (A, F, cost, k) be an input for k-clustering and let $\{C_1^*, \ldots, C_k^*\}$ denote the optimal k-clustering of A with centers $S = \{c_1^*, \ldots c_k^*\}$. Given $\alpha \geq 1$, the instance is *α-perturbation-resilient* if for every cost function cost' on A with

$$\forall\ (x, y) \in A \times F,\ \text{cost}(x, y) \leq \text{cost}'(x, y) \leq \alpha \cdot \text{cost}(x, y),$$

$\{C_1^*, \ldots, C_k^*\}$ is the unique optimal clustering with centers S of the instance (A, F, cost', k).

This notion of stability was historically defined for metric cost functions. We show that a local optimum must be the global optimum in that case. Observe that cost$'$ in Definition 7.3.1 needs not be a metric, even if cost is a metric. Consider a solution S_0 to the k-clustering problem with parameter p. We say that S_0 is $1/\varepsilon$-*locally optimal* if any solution S_1 such that $|S_0 \setminus S_1| + |S_1 \setminus S_0| \leq 2/\varepsilon$ has cost at least cost(S_0).

Theorem 7.3.2. *Let $\alpha > 3 + 2\varepsilon$. For any instance of the k-median problem that is α-perturbation-resilient, any $2(\alpha-3)^{-1}$-locally optimal solution $\mathcal{L}$ induces the optimal clustering $\{C_1^*, \ldots, C_k^*\}$.*

For ease of exposition, we only consider the k-median problem. Applying Lemma 7.2.2 in the proof of Lemma 7.3.4 yields the results for general p with α growing exponentially with p. We define l_x to be the cost for client x in solution $\mathcal{L}$ and g_x to be its cost in the optimal solution S. Finally, for any sets of centers S and $S_0 \subset S$, define $N_S(S_0)$ to be the set of clients served by a center of S_0 in solution S, i.e.: $N_S(S_0) = \{x \mid \exists s \in S_0, \mathrm{dist}(x,s) = \min_{s' \in S} \mathrm{dist}(x,s')\}$.

Define $D = \mathcal{L} \cap S$, $\tilde{\mathcal{L}} = \mathcal{L} \setminus S$, and $\tilde{S} = S \setminus \mathcal{L}$. However, for the remainder of this section except for the proof of Theorem 7.3.3, we will assume $D = \emptyset$ for the sake of a simpler notation.

The proof of Theorem 7.3.2 relies on the following theorem of particular interest.

Theorem 7.3.3 (Local-Approximation Theorem.)**.** *Let $\mathcal{L}$ be a $1/\varepsilon$-locally optimal solution and S be any optimal solution. Then*

$$\sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} l_x \leq (3 + 2\varepsilon) \cdot \sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_x.$$

We first show how Theorem 7.3.3 allows us to prove Theorem 7.3.2.

Proof of Theorem 7.3.2. We can decompose the cost of $\mathcal{L}$ as follows:

$$\begin{aligned} \mathrm{cost}(\mathcal{L}) &= \sum_{\substack{x \in A \setminus \\ (N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}}))}} \mathrm{dist}(x, \mathcal{L}) + \sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} \mathrm{dist}(x, \mathcal{L}) \\ &\leq \sum_{\substack{x \in A \setminus \\ (N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}}))}} \mathrm{dist}(x, S) + (3 + 2\varepsilon) \sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} \mathrm{dist}(x, S) \end{aligned}$$

where the inequality follows from Theorem 7.3.3. Since $\alpha > 3 + 2\varepsilon$, the result follows from α-perturbation resilience. □

We now turn to the proof of Theorem 7.3.3 We first introduce some definitions, following the terminology of [30, 177].

Consider the following bipartite graph $\Gamma = (\tilde{\mathcal{L}} \cup \tilde{S}, \mathcal{E})$ where $\mathcal{E}$ is defined as follows. For any center $f \in \tilde{S}$, we have $(f, \ell) \in \mathcal{E}$ where ℓ is the center of $\tilde{\mathcal{L}}$ that is the closest to f. Denote by $N_\Gamma(\ell)$ the neighbors of the node corresponding to the center ℓ in Γ. For each edge $(f, \ell) \in \mathcal{E}$

and for any client $x \in N_S(\{f\}) \setminus N_{\mathcal{L}}(\{\ell\})$, we define Reassign_x as the cost of reassigning client x to ℓ. We derive the following lemma.

Lemma 7.3.4. *For any client x, $\text{Reassign}_x \leq l_x + 2g_x$.*

Proof. By definition we have $\text{Reassign}_x = \text{dist}(x, \ell)$. By the triangle inequality $\text{dist}(x, \ell) \leq \text{dist}(x, f) + \text{dist}(f, \ell)$. Since f serves x in S, we have $\text{dist}(x, f) = g_x$, hence $\text{dist}(x, \ell) \leq g_x + \text{dist}(f, \ell)$. We now bound $\text{dist}(f, \ell)$. Consider the center ℓ' that serves x in solution $\mathcal{L}$. By the triangle inequality we have $\text{dist}(f, \ell') \leq \text{dist}(x, f) + \text{dist}(x, \ell') = g_x + l_x$. Finally, since ℓ is the closest center of f in $\mathcal{L}$, we have $\text{dist}(f, \ell) \leq \text{dist}(f, \ell') \leq g_x + l_x$ and the lemma follows. □

We partition the centers of $\tilde{\mathcal{L}}$ as follows. Let $\tilde{\mathcal{L}}_0$ be the set of centers of $\tilde{\mathcal{L}}$ that have degree 0 in Γ. Let $\tilde{\mathcal{L}}_{\leq \varepsilon^{-1}}$ be the set of centers of $\tilde{\mathcal{L}}$ that have degree at least one and at most $1/\varepsilon$ in Γ. Let $\tilde{\mathcal{L}}_{>\varepsilon^{-1}}$ be the set of centers of $\tilde{\mathcal{L}}$ that have degree greater than $1/\varepsilon$ in Γ.

We now partition the centers of $\tilde{\mathcal{L}}$ and $\tilde{S}$ using the neighborhoods of the vertices of $\tilde{\mathcal{L}}$ in Γ. We start by iteratively constructing two sets of pairs $E_{\leq \varepsilon^{-1}}$ and $E_{>\varepsilon^{-1}}$. For each center $\ell \in \tilde{\mathcal{L}}_{\leq \varepsilon^{-1}} \cup \tilde{\mathcal{L}}_{>\varepsilon^{-1}}$, we pick a set A_ℓ of $|N_\Gamma(\ell)| - 1$ centers of $\tilde{\mathcal{L}}_0$ and define a pair $(\{\ell\} \cup A_\ell, N_\Gamma(\ell))$. We then remove A_ℓ from $\tilde{\mathcal{L}}_0$ and repeat. Let $E_{\leq \varepsilon^{-1}}$ be the pairs that contain a center of $\tilde{\mathcal{L}}_{\leq \varepsilon^{-1}}$ and let $E_{>\varepsilon^{-1}}$ be the remaining pairs.

The following lemma follows from the definition of the pairs.

Lemma 7.3.5. *Let $(R^{\tilde{\mathcal{L}}}, R^{\tilde{S}})$ be a pair in $E_{\leq \varepsilon^{-1}} \cup E_{>\varepsilon^{-1}}$. If $\ell \in R^{\tilde{\mathcal{L}}}$, then for any f such that $(f, \ell) \in \mathcal{E}$, $f \in R^{\tilde{S}}$.*

Lemma 7.3.6. *For any pair $(R^{\tilde{\mathcal{L}}}, R^{\tilde{S}}) \in E_{\leq \varepsilon^{-1}}$ we have*

$$\sum_{x \in N_S(R^{\tilde{S}})} l_x \leq \sum_{x \in N_S(R^{\tilde{S}})} g_x + 2 \sum_{x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) \setminus N_S(R^{\tilde{S}})} g_x.$$

Proof. Consider the mixed solution $M = \mathcal{L} \setminus R^{\tilde{\mathcal{L}}} \cup R^{\tilde{S}}$. For each point x, let m_x denote the cost of x in solution M. Consider a client from $x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) \setminus N_S(R^{\tilde{S}})$. Since x is served by a center $c_i^* \notin R^{\tilde{S}}$, the center $\ell \in \mathcal{L}$ with $(\ell, c_i^*) \in \Gamma$ is also included in M. Hence, we have

$$m_x \leq \begin{cases} g_x & \text{if } x \in N_S(R^{\tilde{S}}). \\ \text{Reassign}_x & \text{if } x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) \setminus N_S(R^{\tilde{S}}), \text{ see Lemma 7.3.5.} \\ l_x & \text{otherwise.} \end{cases}$$

Now, observe that the solution M differs from $\mathcal{L}$ by at most $1/\varepsilon$ centers. Thus, by $1/\varepsilon$-local optimality we have $\text{cost}(\mathcal{L}) \leq \text{cost}(M)$. Summing over all clients and simplifying, we obtain

$$\sum_{x \in N_S(R^{\tilde{S}}) \cup N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}})} l_x \leq \sum_{x \in N_S(R^{\tilde{S}})} g_x + \sum_{x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) \setminus N_S(R^{\tilde{S}})} \text{Reassign}_x.$$

The lemma follows by combining with Lemma 7.3.4:

$$\begin{aligned} \sum_{x \in N_S(R^{\tilde{S}}) \cup N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}})} l_x &\leq \sum_{x \in N_S(R^{\tilde{S}})} g_x + \sum_{x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) \setminus N_S(R^{\tilde{S}})} l_x + 2g_x \\ \Rightarrow \sum_{x \in N_S(R^{\tilde{S}})} l_x &\leq \sum_{x \in N_S(R^{\tilde{S}})} g_x + 2 \sum_{x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) \setminus N_S(R^{\tilde{S}})} g_x \end{aligned}$$

□

We now analyze the cost of the clients served by a center of $\mathcal{L}$ that has degree greater than ε^{-1} in Γ.

Lemma 7.3.7. *For any pair $(R^{\tilde{\mathcal{L}}}, R^{\tilde{S}}) \in E_{>\varepsilon^{-1}}$ we have*

$$\sum_{x \in N_S(R^{\tilde{S}})} l_x \leq \sum_{x \in N_S(R^{\tilde{S}})} g_x + 2(1+\varepsilon) \sum_{x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}})} g_x.$$

Proof. Consider the center $\hat{\ell} \in R^{\tilde{\mathcal{L}}}$ that has in-degree greater than ε^{-1}. Let $\hat{L} = R^{\tilde{\mathcal{L}}} \setminus \{\hat{\ell}\}$. For each $\ell \in \hat{L}$, we associate a center $f(\ell)$ in $R^{\tilde{S}}$ such that each $f(\ell) \neq f(\ell')$ for $\ell \neq \ell'$. Note that this is possible since $|\hat{L}| = |R^{\tilde{S}}| - 1$. Let $\tilde{f}$ be the center of $R^{\tilde{S}}$ that is not associated with any center of $\hat{L}$.

Now, for each center ℓ of $\hat{L}$, we consider the mixed solution $M^\ell = (\mathcal{L} \setminus \{\ell\}) \cup \{f(\ell)\}$. For each client x, we bound its cost m_x^ℓ in solution M^ℓ. We have

$$m_x^\ell \leq \begin{cases} g_x & \text{if } x \in N_S(\{f(\ell)\}). \\ \text{Reassign}_x & \text{if } x \in N_{\mathcal{L}}(\{\ell\}) \setminus N_S(\{f(\ell)\}), \text{ see Lemma 7.3.5.} \\ l_x & \text{otherwise.} \end{cases}$$

This, we have by local optimality

$$\sum_{x \in N_S(f(\ell)) \cup N_{\mathcal{L}}(\ell))} l_x \leq \sum_{x \in N_S(f(\ell))} g_x + \sum_{x \in N_{\mathcal{L}}(\ell) \setminus N_S(f(\ell))} \text{Reassign}_x.$$

Summing over all centers $\ell \in \hat{L}$ and all the clients in $N_S(\{f(\ell)\}) \cup N_{\mathcal{L}}(\{\ell\})$, we obtain the upper bound

$$\begin{aligned} \sum_{x \in N_S(R^{\tilde{S}} \setminus \{\tilde{f}\}) \cup N_{\mathcal{L}}(\hat{L})} l_x &\leq \sum_{x \in N_S(R^{\tilde{S}} \setminus \{\tilde{f}\})} g_x + \sum_{x \in N_{\mathcal{L}}(\hat{L})} \text{Reassign}_x. \\ \Rightarrow \sum_{x \in N_S(R^{\tilde{S}} \setminus \{\tilde{f}\})} l_x &\leq \sum_{x \in N_S(R^{\tilde{S}} \setminus \{\tilde{f}\})} g_x + \sum_{x \in N_{\mathcal{L}}(\hat{L})} 2g_x. \end{aligned} \tag{7.5}$$

We now complete the proof of the lemma by analyzing the cost of the clients in $N_S(\{\tilde{f}\})$. We consider the center $\ell^* \in \hat{L}$ that minimizes the optimal part of reassignment cost of its

clients, i.e. $\sum_{x \in N_{\mathcal{L}}(\{\ell^*\})} 2g_x$ is minimized by ℓ^* among all centers of $\hat{\mathcal{L}}$. We then consider the solution $M^{(\ell^*,\tilde{f})} = (\mathcal{L} \setminus \{\ell^*\}) \cup \{\tilde{f}\}$. For each client x, we bound its cost $m_x^{(\ell^*,\tilde{f})}$ in solution $M^{(\ell^*,\tilde{f})}$. We have

$$m_x^{(\ell^*,\tilde{f})} \leq \begin{cases} g_x & \text{if } x \in N_S(\{\tilde{f}\}). \\ \text{Reassign}_x & \text{if } x \in N_{\mathcal{L}}(\{\ell^*\}) \setminus N_S(\{\tilde{f}\}), \text{ see Lemma 7.3.5.} \\ l_x & \text{otherwise.} \end{cases}$$

Thus, summing over all clients x, we have by local optimality

$$\begin{aligned} \sum_{x \in N_S(\{\tilde{f}\}) \cup N_{\mathcal{L}}(\{\ell^*\})} l_x &\leq \sum_{x \in N_S(\{\tilde{f}\})} g_x + \sum_{x \in N_{\mathcal{L}}(\{\ell^*\}) \setminus N_S(\{\tilde{f}\})} \text{Reassign}_x \\ \sum_{x \in N_S(\{\tilde{f}\})} l_x &\leq \sum_{x \in N_S(\{\tilde{f}\})} g_x + \sum_{x \in N_{\mathcal{L}}(\{\ell^*\}) \setminus N_S(\{\tilde{f}\})} 2g_x. \end{aligned} \tag{7.6}$$

By our choice of ℓ^*, we have a minimum cost among all $|\hat{\mathcal{L}}| \geq \varepsilon^{-1}$ centers of $\hat{\mathcal{L}}$, that is $\sum_{x \in N_{\mathcal{L}}(\{\ell^*\}) \setminus N_S(\{\tilde{f}\})} g_x \leq \varepsilon \cdot \sum_{x \in N_{\mathcal{L}}(\hat{\mathcal{L}}))} g_x$. Combining this with Equations 7.5 and 7.6, we have

$$\sum_{x \in N_S(R^{\tilde{S}})} l_x \leq \sum_{x \in N_S(R^{\tilde{S}})} g_x + 2(1+\varepsilon) \sum_{x \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}})} g_x.$$

□

We now turn to the proof of Theorem 7.3.3.

Proof of Theorem 7.3.3. Observe first that for any $x \in N_{\mathcal{L}}(\tilde{\mathcal{L}}) \setminus N_S(\tilde{S})$, we have $l_x \leq g_x$. This follows from the fact that the center that serves x in S is in D and so in $\mathcal{L}$ and thus, we have $l_x \leq g_x$. Therefore

$$\sum_{x \in N_{\mathcal{L}}(\tilde{\mathcal{L}}) \setminus N_S(\tilde{S})} l_x \leq \sum_{x \in N_{\mathcal{L}}(\tilde{\mathcal{L}}) \setminus N_S(\tilde{S})} g_x. \tag{7.7}$$

We now sum the equations of Lemmas 7.3.6 and 7.3.7 over all pairs and obtain

$$\sum_{(R^{\tilde{\mathcal{L}}}, R^{\tilde{S}})} \sum_{x \in N_S(R^{\tilde{S}})} l_x \leq \sum_{(R^{\tilde{\mathcal{L}}}, R^{\tilde{S}})} \left(\sum_{x \in N_S(R^{\tilde{S}})} g_x + 2(1+\varepsilon) \sum_{N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}})} g_x \right)$$

$$\Rightarrow \quad \sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} l_x \leq \sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_x + 2(1+\varepsilon) \sum_{x \in N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_x$$

$$\Rightarrow \quad \sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} l_x \leq (3+2\varepsilon) \sum_{x \in N_S(\tilde{S}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_x. \tag{7.8}$$

Combining Equations 7.7 and 7.8 yields the lemma. □

We note that the analysis is essentially tight. Specifically, Local Search with bounded neighborhood size will not, in general, recover the optimal cluster for $(3-\varepsilon)$-perturbation resilient instances.

Proposition 7.3.8. *For any sufficiently small constant $\varepsilon > 0$, there exists an infinite family of $3-\varepsilon$-perturbation-resilient instances such that there is a locally optimal solution that has cost at least $3OPT$.*

Proof. Consider a tripartite graph with nodes O, C, and L, where O is the set of optimal centers, L is the set of centers of a locally optimal solution, and C is the set of clients. We have $|O| = |L| = k$ and $|C| = k^2$. We specify the distances as follows. First, assume some arbitrary but fixed ordering on the elements of O, L, and C. Then $\text{dist}(O_i)(C_{i,j}) = 1+\varepsilon/3$ and $\text{dist}(L_i)(C_{j,i}) = 3$ for any $i, j \in [k]$. All other distances are induced by the shortest path metric along the edges of the graph, i.e. $\text{dist}(O_i)(C_{j,\ell}) = 7+\varepsilon/3$ and $\text{dist}(L_i)(C_{j,\ell}) = 5+2\varepsilon/3$ for $j, \ell \neq i$. We first note that O is indeed the optimal solution with a cost of $k^2 \cdot (1+\varepsilon/3)$. Multiplying the distances $\text{dist}(O_i)(C_{i,j})$ by a factor of $(3-\varepsilon)$ for all $i \in [k]$ and $j \bmod k = i$, still ensures that O is an optimal solution with a cost of $k^2 \cdot (1+\varepsilon/3) \cdot (3-\varepsilon) = k^2 \cdot 3(1-\varepsilon^2)$, which shows that the instance is $(3-\varepsilon)$-perturbation resilient.

What remains to be shown is that L is locally optimal. Assume that we swap out s centers. Due to symmetry, we can consider the solution $\{O_i | i \in [s]\} \cup \{L_i | i \in [k] \setminus [s]\}$. Each of the centers $\{O_i | i \in [s]\}$ serves k clients with a total cost of $k \cdot s \cdot (1+\varepsilon/3)$. The remaining clients are served by $\{L_i | i \in [k] \setminus [s]\}$, as $5+2\varepsilon/3 < 7+\varepsilon/3$. The cost amounts to $s \cdot (k-s) \cdot (5+2\varepsilon/3)$ for the clients that get reassigned and $(k-s)^2 \cdot 3$ for the remaining clients. Combining these three figures gives us a cost of $k^2 \cdot 3 + ks\varepsilon - s^2 \cdot (2+2\varepsilon/3) > k^2 \cdot 3 + ks\varepsilon - s^2 \cdot 3$. For $k > \frac{3s}{\varepsilon}$, this is greater than $k^2 3$, the cost of L. □

7.4 Spectral Separability

In this section, we focus on the third and final stability condition, called "spectral separation".

Definition 7.4.1 (Spectral Separation [222][a])**.** Let $(A, \mathbb{R}^d, ||\cdot||^2, k)$ be an input for k-means clustering in Euclidean space and let $\{C_1^*, \dots C_k^*\}$ denote an optimal clustering of A with centers $S = \{c_1^*, \dots c_k^*\}$. Denote by C an $n \times d$ matrix such that the row $C_i = \underset{c_j^* \in S}{\text{argmin}} \, ||A_i - c_j^*||^2$. Denote by $||\cdot||_2$ the spectral norm of a matrix. Then $\{C_1^*, \dots C_k^*\}$ is *γ-spectrally separated*, if for any pair (i, j) the following condition holds:

$$||c_i^* - c_j^*|| \geq \gamma \cdot \left(\frac{1}{\sqrt{|C_i^*|}} + \frac{1}{\sqrt{|C_j^*|}} \right) ||A - C||_2.$$

[a] The proximity condition of Kumar and Kannan [222] implies the spectral separation condition.

Since this stability is defined over non-finite metric spaces, we require standard preprocessing steps in order to use Local Search, see Algorithm 11. They consist of reducing the number of dimensions and discretizing the space in order to bound the number of candidate centers.

Algorithm 11 Project and Local Search

1: Project points A onto the best rank k/ε subspace
2: Embed points into a random subspace of dimension $O(\varepsilon^{-2} \log n)$
3: Compute candidate centers (Corollary 7.2.19)
4: Local Search($O(\varepsilon^{-7})$)
5: Output clustering

Theorem 7.4.2. *Let $(A, \mathbb{R}^d, ||\cdot||^2, k)$ be an instance of Euclidean k-means clustering with optimal clustering $C = \{C_1^*, \dots C_k^*\}$ and centers $S = \{c_1^*, \dots c_k^*\}$. If C is more than $3\sqrt{k}$-spectrally separated then Algorithm 11 is a polynomial time approximation scheme.*

In previous work by Kumar and Kannan [222], an algorithm was given with approximation ratio $1 + O(\mathrm{OPT}_k/\mathrm{OPT}_{k-1})$, where OPT_i denotes the value of an optimal solution using i centers. If we assume $\mathrm{OPT}_k/\mathrm{OPT}_{k-1} \leq \varepsilon$, then the optimal k-clustering C is $\Omega(\sqrt{k/\varepsilon})$-spectrally separated [222]. Thus our assumption in Theorem 7.4.2 that C is $3\sqrt{k}$-spectrally separated is weaker (it does not depend on ε) and therefore, our result is stronger since the approximation guarantee does not depend on the assumption about instances. We note that the previous algorithms focused on recovering the optimal target clustering and not optimizing the k-means objective function, though there exists some overlap. In general, a $(1+\varepsilon)$-approximation does not have to agree with the target clustering on a majority of points. There are applications where finding the correct classification is more relevant than minimizing the value of the k-means objective and vice versa.

Nowadays, a standard preprocessing step in Euclidean k-means clustering is to project A onto the subspace spanned by the rank k-approximation. Indeed, this is the first step of the algorithm by Kumar and Kannan [222] (see Algorithm 12).

Algorithm 12 k-means with spectral initialization [222]

1: Project points onto the best rank k subspace
2: Compute a clustering C with constant approximation factor on the projection
3: Initialize centroids of each cluster of C as centers in the original space
4: Run Lloyd's k-means until convergence

In general, projecting onto the best rank k subspace and computing a constant approximation on the projection results in a constant approximation in the original space. Kumar and Kannan [222] and later Awasthi and Sheffet [37] gave tighter bounds if the spectral separation is large enough. Our algorithm omits steps 3 and 4. Instead, we project onto slightly more dimensions in step 1 and subsequently use Local Search as the constant factor approximation

in step 2. To utilize Local Search, we further require a candidate set of solutions, which is described in Section 7.2.1.

It is easy to show that spectral separability implies distribution stability if the dimension is of order k: (1) the distance between centers is $\Omega(\left(\frac{1}{\sqrt{|C_i^*|}} + \frac{1}{\sqrt{|C_j^*|}}\right)\sqrt{k} \cdot ||A - C||_2) = \Omega\left(\left(\frac{1}{\sqrt{|C_i^*|}} + \frac{1}{\sqrt{|C_j^*|}}\right)\sqrt{\mathrm{OPT}}\right)$, and (2) the distance of any point to the "wrong" center is at least $1/2$ of this amount, i.e. the cost of assigning a point to the "wrong" cluster is $\Omega(\frac{\mathrm{OPT}}{|C_j|})$. Projecting onto sufficiently many dimensions allows us to transform a high dimensional point set into a low dimensional one, see, for instance, recent work by Cohen et al. [109]. The projection retains the cost [109] and spectral separability of a clustering, however it does not preserve optimality. In particular, the distance of a single point to the centroids of the clusters can be arbitrarily distorted (see Figure 7.3), which prevents us from using the naive reduction to distribution stability.

Instead, we locally improve the optimal clustering by reassigning points (Lemma 7.4.3). A large contraction of relevant distances can only happen for few points, i.e. the cluster sizes are roughly the same. For the remaining points, we can show that they are guaranteed to have a minimum distance to the wrong center.

Recall that the clustering matrix $X \in \mathbb{R}^{n \times k}$ of a clustering $C = \{C_1, \dots C_k\}$ is defined as $X_{i,j} = \frac{1}{\sqrt{|C_j|}}$ if $A_i \in C_j$ and $X_{i,j} = 0$ otherwise. The following crucial lemma relates spectral separation and distribution stability.

Lemma 7.4.3. *For a point set A, let $C = \{C_1, \dots, C_k\}$ be an optimal clustering with centers $S = \{c_1, \dots, c_k\}$ and associated clustering matrix X. C is at least $\gamma \cdot \sqrt{k}$ spectrally separated with $\gamma > 3$. For $\varepsilon > 0$, let A_m be the best rank $m = k/\varepsilon$ approximation of A. Then there exists a clustering $C_m = \{C_1', \dots, C_k'\}$ and a set of centers S_m, such that*

1. *the cost of clustering A_m with centers S_m via the assignment of C_m is less than $||A_m - XX^T A_m||_F^2$ and*
2. *(C_m, S_m) is $\Omega((\gamma - 3)^2 \cdot \varepsilon)$-distribution stable.*

We note that this lemma would also allow us to use the PTAS of Awasthi et al. [34]. Before giving the proof, we outline how Lemma 7.4.3 helps us prove Theorem 7.4.2. We first notice that if the rank of A is of order k then elementary bounds on matrix norm show that spectral separability implies distribution stability. We combine this observation with the following theorem due to Cohen et al. [110], see also the related earlier work by Feldman et al. [147]. Informally, it states that for every rank k approximation, (and in particular for every constrained rank k approximation such as k-means clustering), projecting to the best rank k/ε subspace is cost-preserving.

Theorem 7.4.4 (Theorem 7 of [110]). *For any $A \in \mathbb{R}^{n\times d}$, let A' be the rank $\lceil k/\varepsilon \rceil$-approximation of A. Then there exists some positive number c such that for any rank k orthogonal projection P,*

$$||A - PA||_F^2 \leq ||A' - PA'||_F^2 + c \leq (1+\varepsilon)||A - PA||_F^2.$$

In the following, we denote $A_m = U_m \Sigma_m V_m^T$ as the rank m approximation of A, where U_m contains the first m left singular vectors of A, Σ_m is the truncated diagonal matrix containing the first m singular values of A and is 0 otherwise, and V_m contains the first m right singular vectors of A. We repeat the theorem as given in the reference. It is perhaps informative to know that the positive number c of the theorem is $||A - A_m||_F^2$.

The combination of the low rank case and this theorem is not trivial as points may be closer to a wrong center after projecting, see also Figure 7.3. Lemma 7.4.3 determines the existence of a clustering whose cost for the projected points A_m is at most the cost of C. Moreover, this clustering has constant distribution stability as well which, combined with the results from Section 7.2.1, allows us to use Local Search. Given that we can find a clustering with cost at most $(1+\varepsilon) \cdot ||A_m - XX^T A_m||_F^2$, Theorem 7.4.4 implies that we will have a $(1+\varepsilon)^2$-approximation overall.

To prove the lemma, we require the following steps:

- Determine a lower bound on the distance of the projected centers $||c_i^T V_m V_m^T - c_j^T V_m V_m^T|| \approx ||c_i - c_j||$.
- Find a clustering C_m with centers $S_m^* = \{c_1^{*T} V_m V_m^T, \ldots, c_k^{*T} V_m V_m^T\}$ of A_m with cost less than $||A_m - XX^T A_m||_F^2$.
- Show that in a well-defined sense, C_m and C agree on a large fraction of points.
- For any point $x \in (C_m)_i$, show that the distance of x to any center not associated with $(C_m)_i$ is large.

We first require a technical statement.

Lemma 7.4.5. *For a point set A, let $C = \{C_1, \ldots C_k\}$ be a clustering with associated clustering matrix X and let A' and A'' be optimal low rank approximations where $k \leq rank(A') < rank(A'')$ holds without loss of generality. Then for each cluster C_i*

$$\left|\left| \frac{1}{|C_i|} \sum_{j \in C_i} \left(A_j'' - A_j'\right) \right|\right|_2 \leq \sqrt{\frac{k}{|C_i|}} \cdot ||A - XX^T A||_2.$$

Proof. By Fact 2.5.2 $|C_i| \cdot ||\frac{1}{|C_i|} \sum_{j \in C_i} (A_i'' - A_i')||_2^2$ is, for the points in C_i, the cost of moving the centroid of the cluster computed on A'' to the centroid of the cluster computed on A'.

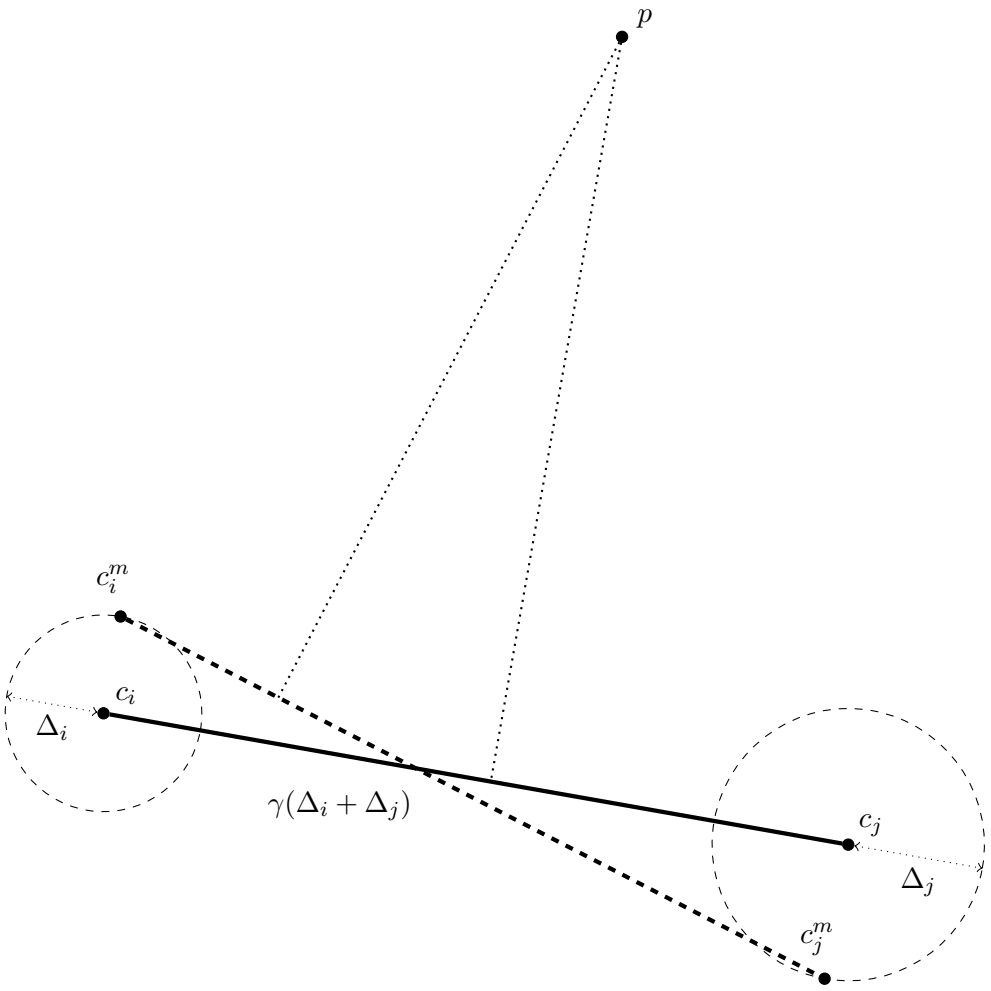

Figure 7.3: Despite the centroids of each cluster being close after computing the best rank m approximation, the projection of a point p to the line connecting the centroid of cluster C_i and C_j can change after computing the best rank m approximation. In this case $||p - c_j|| < ||p - c_i||$ and $||p - c_i^m|| < ||p - c_j^m||$.

For a clustering matrix X, $||XX^TA'' - XX^TA'||_F^2$ is the sum of squared distances of moving the centroids computed on the point set A'' to the centroids computed on A'. We then have

$$\begin{aligned} |C_i| \cdot \left|\left| \frac{1}{|C_i|} \sum_{j \in C_i} (A''_j - A'_j) \right|\right|_2^2 &\leq ||XX^TA'' - XX^TA'||_F^2 \leq ||XX^T||_F^2 \cdot ||A'' - A'||_2^2 \\ &= ||X||_F^2 \cdot ||A'' - A'||_2^2 \leq k \cdot \sigma_{k+1}^2 \leq k \cdot ||A - XX^TA||_2^2. \end{aligned}$$

□

Proof of Lemma 7.4.3. For any point p associated with some row of A, let $p^m = pV_mV_m^T$ be the corresponding row in A_m. Similarly, for some cluster C_i, denote the center in A by c_i and the center in A_m by c_i^m. Extend these notion analogously for projections p^k and c_i^k to the span of the best rank k approximation A_k.

We have for any $m \geq k$ and $i \neq j$

$$\begin{aligned} ||c_i^m - c_j^m|| &\geq ||c_i - c_j|| - ||c_i - c_i^m|| - ||c_j - c_j^m|| \\ &\geq \gamma \cdot \left(\frac{1}{\sqrt{|C_i|}} + \frac{1}{\sqrt{|C_j|}} \right) \sqrt{k} ||A - XX^TA||_2 \\ &\quad - \frac{1}{\sqrt{|C_i|}} \sqrt{k} ||A - XX^TA||_2 - \frac{1}{\sqrt{|C_j|}} \sqrt{k} ||A - XX^TA||_2 \\ &= (\gamma - 1) \cdot \left(\frac{1}{\sqrt{|C_i|}} + \frac{1}{\sqrt{|C_j|}} \right) \sqrt{k} ||A - XX^TA||_2, \end{aligned} \tag{7.9}$$

where the second inequality follows from Lemma 7.4.5.

In the following, let $\Delta_i = \frac{\sqrt{k}}{\sqrt{|C_i|}} ||A - XX^TA||_2$. We will now construct our target clustering C_m. Note that we require this clustering (and its properties) only for the analysis. We distinguish between the following three cases.

Case 1: $p \in C_i$ **and** $c_i^m = \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^m - c_j^m||$**:**

These points remain assigned to c_i^m. The distance between p^m and a different center c_j^m is at least $\frac{1}{2}||c_i^m - c_j^m|| \geq \frac{\gamma-1}{2}(\Delta_i + \Delta_j)$ due to Equation 7.9.

Case 2: $p \in C_i$**,** $c_i^m \neq \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^m - c_j^m||$**, and** $c_i^k \neq \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^k - c_j^k||$**:**

We reassign these points to their closest center.

The distance between p^m and a different center c_j^m is at least $\frac{1}{2}||c_i^m - c_j^m|| \geq \frac{\gamma-1}{2}(\Delta_i + \Delta_j)$ due to Equation 7.9.

Case 3: $p \in C_i$**,** $c_i^m \neq c_h^m = \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^m - c_j^m||$**, and** $c_i^k = \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^k - c_j^k||$**:**

We assign p^m to c_h^m at the cost of a slightly weaker movement bound on the distance between p^m and c_j^m. Due to orthogonality of V, we have for $m > k$, $(V_m - V_k)^T V_k = V_k^T(V_m - V_k) = 0$. Hence $V_m V_m^T V_k = V_m V_k^T V_k + V_m(V_m - V_k)^T V_k = V_k V_k^T V_k + (V_m - V_k)V_k^T V_k = V_k V_k^T V_k = V_k$. Then $p^k = p^T V_k V_k^T = p^T V_m V_m^T V_k V_k^T = (p^m)^T V_k V_k^T$.

Further, $||p^k - c_h^k|| \geq \frac{1}{2}||c_h^k - c_i^k|| \geq \frac{\gamma-1}{2}(\Delta_i + \Delta_h)$ due to Equation 7.9. Then the distance between p^m and a different center c_j^m

$$\begin{aligned} ||p^m - c_j^m|| &\geq ||p^m - c_j^k|| - ||c_j^m - c_j^k|| = \sqrt{||p^m - p^k||^2 + ||p^k - c_j^k||^2} - ||c_j^m - c_j^k|| \\ &\geq ||p^k - c_j^k|| - \Delta_j \geq \frac{\gamma - 3}{2}(\Delta_i + \Delta_j), \end{aligned}$$

where the equality follows from orthogonality and the second to last inequality follows from Lemma 7.4.5.

Now, given the centers $\{c_1^m, \dots c_k^m\}$, we obtain a center matrix M_{C_m} where the ith row of M_{C_m} is the center according to the assignment above. Since both clusterings use the same centers but C_m improves locally on the assignments, we have $||A_m - M_{C_m}||_F^2 \leq ||A_m - XX^T A_m||_F^2$, which proves the first statement of the lemma. Additionally, due to the fact that $A_m - XX^T A_m$ has rank $m = k/\varepsilon$, we have

$$||A_m - M_{C_m}||_F^2 \leq ||A_m - XX^T A_m||_F^2 \leq m \cdot ||A_m - XX^T A_m||_2^2 \leq k/\varepsilon \cdot ||A - XX^T A||_F^2 \quad (7.10)$$

To ensure stability, we show that for each element of C_m there exists an element of C, such that both clusters agree on a large fraction of points. We prove this by using techniques from Awasthi and Sheffet [37] (Theorem 3.1) and Kumar and Kannan [222] (Theorem 5.4), which we repeat for completeness.

Lemma 7.4.6. *Let $C_m = \{C_1', \dots C_k'\}$ and $C = \{C_1, \dots C_k\}$ be defined as above. Then there exists a bijection $b : C \to C_m$ such that for any $i \in \{i, \dots, k\}$*

$$\left(1 - \frac{32}{(\gamma-1)^2}\right)|C_i| \leq b(|C_i|) \leq \left(1 + \frac{32}{(\gamma-1)^2}\right)|C_i|.$$

Proof. Denote by $T_{i \to j}$ the set of points from C_i such that $||c_i^k - p^k|| > ||c_j^k - p^k||$. We first note that rank of the matrix $A_k - XX^T A$ is at most $2k$ as the rank of A_k is at most k and the rank of $XX^T A$ is at most k. Further A_k is the minimizer among all rank k approximations, i.e. in particular $||A - A_k||_F \leq ||A - XX^T A||_F$. Then $||A_k - XX^T A||_F^2 \leq 2k \cdot ||A_k - XX^T A||_2^2 \leq 2k \cdot \left(||A - A_k||_2 + ||A - XX^T A||_2\right)^2 \leq 8k \cdot ||A - XX^T A||_2^2 \leq 8 \cdot |C_i| \cdot \Delta_i^2$ for any $i \in \{1, \dots, k\}$. We can lower bound the distance $||p^k - c_i^k|| \geq \frac{1}{2}||c_i^k - c_j^k|| \geq \frac{\gamma-1}{2} \cdot \left(\frac{1}{\sqrt{C_i}} + \frac{1}{\sqrt{|C_j|}}\right)\sqrt{k}||A - XX^T A||_2^2$. Assigning these points to c_i^k, we can bound the total number of points added to and subtracted

from cluster C_j by observing

$$\Delta_j^2 \sum_{i \neq j} |T_{i \to j}| \leq \sum_{i \neq j} |T_{i \to j}| \cdot \left(\frac{\gamma - 1}{2}\right)^2 \cdot (\Delta_i + \Delta_j)^2 \leq ||A_k - XX^T A||_F^2 \leq 8 \cdot |C_j| \cdot \Delta_j^2$$
$$\Delta_j^2 \sum_{i \neq j} |T_{j \to i}| \leq \sum_{j \neq i} |T_{j \to i}| \cdot \left(\frac{\gamma - 1}{2}\right)^2 \cdot (\Delta_i + \Delta_j)^2 \leq ||A_k - XX^T A||_F^2 \leq 8 \cdot |C_j| \cdot \Delta_j^2.$$

Therefore, the cluster sizes are up to some multiplicative factor of $\left(1 \pm \frac{32}{(\gamma-1)^2}\right)$ identical. □

We now have for each point $p^m \in C_i'$ a minimum cost of

$$\begin{aligned} ||p^m - c_j^m||^2 &\geq \left(\frac{\gamma - 3}{2} \cdot \left(\frac{1}{\sqrt{|C_i|}} + \frac{1}{\sqrt{|C_j|}}\right) \cdot \sqrt{k} \cdot ||A - XX^T A||_2\right)^2 \\ &\geq \left(\frac{\gamma - 3}{2} \cdot \left(\sqrt{\frac{1}{\left(1 + \frac{32}{(\gamma-1)^2}\right) \cdot |C_i'|}} + \sqrt{\frac{1}{\left(1 + \frac{32}{(\gamma-1)^2}\right) \cdot |C_j'|}}\right) \cdot \sqrt{k} \cdot ||A - XX^T A||_2\right)^2 \\ &\geq \frac{(\gamma - 3)^2}{36} \cdot \varepsilon \frac{||A_m - M_{C_m}||_F^2}{|C_j'|} \end{aligned}$$

where the first inequality holds due to Case 3, the second inequality holds due to Lemma 7.4.6 and the last inequality follows from $\gamma > 3$ and Equation 7.10. This ensures that the distribution stability condition is satisfied. □

We conclude by proving our main theorem of the section.

Proof of Theorem 7.4.2. Let $m > 7k/\varepsilon$ and let A_m be the best rank m approximation of A. Given the optimal clustering C^* of A with clustering matrix X, Lemma 7.4.3 guarantees the existence of a clustering C_m with center matrix M_{C_m} such that $||A_m - M_{C_m}||_F^2 \leq ||A_m - XX^T A_m||$ and that C has constant distribution stability. Using the previous lemmas of this section, we can show that we can find a clustering of A_m with cost at most $(1 + \varepsilon/7)$ times the cost of an optimal k-means clustering of A_m. Due to Corollary 7.2.19, there exists a discretization $(A_m, F, ||\cdot||^2, k)$ of $(A_m, \mathbb{R}^d, ||\cdot||^2, k)$ such that the clustering C of the first instance has at most $(1 + \varepsilon/7)$ times the cost of C in the second instance and such that C has constant distribution stability. By Theorem 7.2.4, Local Search with appropriate (but constant) neighborhood size will find a clustering C' with cost at most $(1 + \varepsilon/7)$ times the cost of C_m in $(A_m, F, ||\cdot||^2, k)$. Let Y be the clustering matrix of C'. We then have $||A_m - YY^T A_m||_F^2 + ||A - A_m||_F^2 \leq (1+\varepsilon/7)^2 ||A_m - M_{C_m}||_F^2 + ||A - A_m||_F^2 \leq (1+\varepsilon/7)^2 ||A_m - XX^T A_m||_F^2 + ||A - A_m||_F^2 \leq (1 + \varepsilon/7)^3 ||A - XX^T A||_F^2 \leq (1 + \varepsilon)\text{OPT}$, where the second to last inequality is due to Theorem 7.4.4. □

8 Conclusion and Open Questions

We would like to conclude with a summary of the main contributions and open questions.

Matching in Datastreams We have investigated algebraic techniques for estimating matching sizes in dynamic data streams. Barring tight bounds on the size of $O(1)$-Ruzsa-Seméredi graphs, which does not seem to be within the scope of current available techniques, the problem is settled, both for estimating matching sizes and for computing the matching. The biggest open question is whether it is possible to improve on the 2-approximation ratio in insertion-only streams for either problem. Konrad et al. [220] showed that 2 passes are already sufficient to compute an improved matching.

Algorithms based on augmenting paths are difficult to use in a space constrained setting. For estimating the matching size, the aforementioned algebraic approaches, or perhaps the fractional matching might be a viable alternative avenue.

Nearest Neighbor Searching in Datastreams Our nearest neighbor filtering algorithm was the first to give any form of guarantee in dynamic data streams. The presented algorithm was merely a first step to explore the field, and we expect many improvements to be possible for various similarity measures and distance functions.

We also previously noted that most locality sensitive hashing approaches are readily implementable in insertion-only streams due to the fact that they are *data independent*. Recently, Andoni et al. [18, 22] introduced data dependent locality sensitive hashing. While these approaches have better guarantees, they are also less suited to streaming environments. It would be interesting to study if, and to what degree, data dependent hashing can be performed in insertion-only data streams.

Clustering in Sliding Windows We gave an optimal algorithm for estimating the diameter and for solving the 2-center problem. For an arbitrary number of centers, we have a gap between the lower bound of 4 and the upper bound of 6. A natural question is to ask what the correct answer is. At a first glance, we might expect an improvement over the 6 to be possible. However, a (minor) gap between 2-center and k-center is also observable in insertion-only streams. For the former, obtaining a factor 2 is possible by simply using the first point p as a center and as a second center the point furthest away from p. For the latter, Guha [170] showed a lower bound of $\Omega(k^2)$ points for any algorithm obtaining an estimation of

the objective function beyond a factor of $2 + 1/k$. Since all known algorithms for the k-center problem in streams require an estimation of the objective value if $k > 2$, they are subject to this lower bound.

In our case, we do not obtain a 4-approximation to the objective function for 2-center, despite being able to prove that the clustering cost is a 4-approximation. Our techniques for obtaining lower bounds cannot prove a factor of 6, as they give an estimate of the objective value for "free". It is unclear to the author, whether these difficulties are merely an artifact of the analysis, or whether they hint at a perhaps surprising gap between 2- and 3-center in sliding windows.

Jaccard Center We gave a hardness results, as well as a PTAS and showed that the problem is in FPT. We expect that the analysis may be extended to other rational set similarities. A more interesting open question is whether the problem admits an efficient PTAS, i.e. a PTAS with running time $\text{poly}(n, |U|) \cdot \exp(\text{poly}(\varepsilon^{-1}))$. Our initial reason for studying fixed parameter tractability was that if a problem is not in FPT, it does not admit an efficient PTAS [83].

Another question is whether it is possible to improve over the 2-approximation ratio for 2-center with Jaccard distance. It is possible to obtain a PTAS for 2-center in unconstrained Euclidean spaces [45]. However, despite the embeddability of the Jaccard distance into Euclidean spaces, this algorithm cannot be used for the constrained set of centers of corresponding to item sets.

Clustering via Local Search We gave evidence why Local Search preforms well in practice by showing that for most stability conditions, Local Search can compute optimal or near optimal clusterings. An interesting question is whether we can make similar observation for other clustering tasks such as hierarchical clustering and density clustering. While we do not believe that Local Search is particularly suited for any of these problems, there exist a number of proposed heuristics such as single or complete linkage.

Another question is whether this type of analysis can be done for other problems which are hard to solve in the worst-case, but where Local Search has reasonable performance in practice. The classic traveling salesperson problem may be one such candidate, see e.g. Chapter 8 in Aarts and Lenstra [1].

Bibliography

[1] Aarts, E., and Lenstra, J. K., Eds. *Local Search in Combinatorial Optimization*, 1st ed. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[2] Achlioptas, D., and McSherry, F. On spectral learning of mixtures of distributions. In *Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings* (2005), pp. 458–469.

[3] Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. Approximating extent measures of points. *J. ACM 51*, 4 (2004), 606–635.

[4] Agarwal, P. K., Matousek, J., and Suri, S. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom. 1* (1991), 189–201.

[5] Agarwal, P. K., and Sharathkumar, R. Streaming algorithms for extent problems in high dimensions. *Algorithmica 72*, 1 (2015), 83–98.

[6] Agarwal, P. K., and Yu, H. A space-optimal data-stream algorithm for coresets in the plane. In *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007* (2007), pp. 1–10.

[7] Ahmadian, S., Norouzi-Fard, A., Svensson, O., and Ward, J. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *CoRR abs/1612.07925* (2016).

[8] Ahn, K. J., Cormode, G., Guha, S., McGregor, A., and Wirth, A. Correlation clustering in data streams. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (2015), pp. 2237–2246.

[9] Ahn, K. J., and Guha, S. Graph sparsification in the semi-streaming model. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)* (2009), pp. 328–338.

[10] Ahn, K. J., and Guha, S. Linear programming in the semi-streaming model with application to the maximum matching problem. *Information and Computation 222* (2013), 59–79.

[11] Ahn, K. J., Guha, S., and McGregor, A. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2012), pp. 459–467.

[12] Ahn, K. J., Guha, S., and McGregor, A. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)* (2012), pp. 5–14.

[13] Ai, Y., Hu, W., Li, Y., and Woodruff, D. P. New characterizations in turnstile streams with applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan* (2016), pp. 20:1–20:22.

[14] Ailon, N., Jaiswal, R., and Monteleoni, C. Streaming k-means approximation. In *NIPS* (2009), pp. 10–18.

[15] Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences 58*, 1 (1999), 137–147.

[16] Alon, N., Moitra, A., and Sudakov, B. Nearly complete graphs decomposable into large induced matchings and their applications. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012* (2012), pp. 1079–1090.

[17] Andoni, A., and Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM 51*, 1 (2008), 117–122.

[18] Andoni, A., Indyk, P., Nguyen, H. L., and Razenshteyn, I. P. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014* (2014), pp. 1018–1028.

[19] Andoni, A., Indyk, P., and Patrascu, M. On the optimality of the dimensionality reduction method. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings* (2006), pp. 449–458.

[20] Andoni, A., and Nguyen, H. L. Width of points in the streaming model. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2012), pp. 447–452.

[21] Andoni, A., and Nguyen, H. L. Eigenvalues of a matrix in the streaming model. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2013), pp. 1729–1737.

[22] Andoni, A., and Razenshteyn, I. P. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015* (2015), pp. 793–801.

[23] Ardjmand, E., Park, N., Weckman, G., and Amin-Naseri, M. R. The discrete unconscious search and its application to uncapacitated facility location problem. *Computers & Industrial Engineering 73* (2014), 32 – 40.

[24] Arora, S., and Kannan, R. Learning mixtures of arbitrary gaussians. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece* (2001), pp. 247–257.

[25] Arora, S., Raghavan, P., and Rao, S. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998* (1998), pp. 106–113.

[26] Arthur, D., Manthey, B., and Röglin, H. Smoothed analysis of the k-means method. *J. ACM 58*, 5 (2011), 19.

[27] Arthur, D., and Vassilvitskii, S. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007* (2007), pp. 1027–1035.

[28] Arthur, D., and Vassilvitskii, S. Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method. *SIAM J. Comput. 39*, 2 (2009), 766–782.

[29] Arya, S., and Chan, T. M. Better ε-dependencies for offline approximate nearest neighbor search, euclidean minimum spanning trees, and ε-kernels. In *30th Annual Symposium on Computational Geometry, SOCG'14, Kyoto, Japan, June 08 - 11, 2014* (2014), p. 416.

[30] Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. Local search heuristics for k-median and facility location problems. *SIAM J. Comput. 33*, 3 (2004), 544–562.

[31] Assadi, S., Khanna, S., and Li, Y. Tight bounds for single-pass streaming complexity of the set cover problem. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016* (2016), pp. 698–711.

[32] Assadi, S., Khanna, S., and Li, Y. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19* (2017), pp. 1723–1742.

[33] Assadi, S., Khanna, S., Li, Y., and Yaroslavtsev, G. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016* (2016), pp. 1345–1364.

[34] Awasthi, P., Blum, A., and Sheffet, O. Stability yields a PTAS for k-median and k-means clustering. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA* (2010), pp. 309–318.

[35] Awasthi, P., Blum, A., and Sheffet, O. Center-based clustering under perturbation stability. *Inf. Process. Lett. 112*, 1-2 (2012), 49–54.

[36] Awasthi, P., Charikar, M., Krishnaswamy, R., and Sinop, A. K. The hardness of approximation of Euclidean k-means. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands* (2015), pp. 754–767.

[37] Awasthi, P., and Sheffet, O. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings* (2012), pp. 37–49.

[38] Babcock, B., Datar, M., Motwani, R., and O'Callaghan, L. Maintaining variance and k-medians over data stream windows. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA* (2003), pp. 234–243.

[39] Bachmaier, C., Brandenburg, F. J., Gleissner, A., and Hofmeier, A. On the hardness of maximum rank aggregation problems. *J. Discrete Algorithms 31* (2015), 2–13.

[40] Bachrach, Y., and Herbrich, R. Fingerprinting ratings for collaborative filtering - theoretical and empirical analysis. In *String Processing and Information Retrieval - 17th International Symposium, SPIRE 2010, Los Cabos, Mexico, October 11-13, 2010. Proceedings* (2010), pp. 25–36.

[41] Bachrach, Y., and Porat, E. Fingerprints for highly similar streams. *Inf. Comput. 244* (2015), 113–121.

[42] Backurs, A., Indyk, P., Razenshteyn, I. P., and Woodruff, D. P. Nearly-optimal bounds for sparse recovery in generic norms, with applications to k-median sketching. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016* (2016), pp. 318–337.

[43] Badoiu, M., and Clarkson, K. L. Smaller core-sets for balls. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.* (2003), pp. 801–802.

[44] Badoiu, M., and Clarkson, K. L. Optimal core-sets for balls. *Comput. Geom. 40*, 1 (2008), 14–22.

[45] Badoiu, M., Har-Peled, S., and Indyk, P. Approximate clustering via core-sets. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada* (2002), pp. 250–257.

[46] Balcan, M., Blum, A., and Gupta, A. Approximate clustering without the approximation. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009* (2009), pp. 1068–1077.

[47] Balcan, M., Blum, A., and Gupta, A. Clustering under approximation stability. *J. ACM 60*, 2 (2013), 8.

[48] Balcan, M., and Braverman, M. Finding low error clusterings. In *COLT 2009 - The 22nd Conference on Learning Theory, Montreal, Quebec, Canada, June 18-21, 2009* (2009).

[49] Balcan, M., Haghtalab, N., and White, C. k-center clustering under perturbation resilience. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy* (2016), pp. 68:1–68:14.

[50] Balcan, M., and Liang, Y. Clustering under perturbation resilience. *SIAM J. Comput. 45*, 1 (2016), 102–155.

[51] Balcan, M., Röglin, H., and Teng, S. Agnostic clustering. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings* (2009), pp. 384–398.

[52] Bandyapadhyay, S., and Varadarajan, K. R. On variants of k-means clustering. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA* (2016), pp. 14:1–14:15.

[53] Bar-Yossef, Z., Jayram, T. S., and Kerenidis, I. Exponential separation of quantum and classical one-way communication complexity. *SIAM Journal on Computing 38*, 1 (2008), 366–384.

[54] Bar-Yossef, Z., Kumar, R., and Sivakumar, D. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2002), pp. 623–632.

[55] Baswana, S. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Information Processing Letters 106*, 3 (2008), 110–114.

[56] Bateni, M., Bhaskara, A., Lattanzi, S., and Mirrokni, V. S. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada* (2014), pp. 2591–2599.

[57] Belkin, M., and Sinha, K. Polynomial learning of distribution families. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA* (2010), pp. 103–112.

[58] Ben-David, S., Chor, B., Goldreich, O., and Luby, M. On the theory of average case complexity. *Journal of Computer and system Sciences 44*, 2 (1992), 193–219.

[59] Ben-David, S., and Reyzin, L. Data stability in clustering: A closer look. *Theor. Comput. Sci. 558* (2014), 51–61.

[60] Bentley, J. L., and Saxe, J. B. Decomposable searching problems i: Static-to-dynamic transformation. *J. Algorithms 1*, 4 (1980), 301–358.

[61] Bhattacharya, S., Henzinger, M., Nanongkai, D., and Tsourakakis, C. E. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015* (2015), pp. 173–182.

[62] Biedl, T. C., Brandenburg, F., and Deng, X. On the complexity of crossings in permutations. *Discrete Mathematics 309*, 7 (2009), 1813–1823.

[63] Bilu, Y., Daniely, A., Linial, N., and Saks, M. E. On the practically interesting instances of MAXCUT. In *30th International Symposium on Theoretical Aspects of*

Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany (2013), pp. 526–537.

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

[65] Blelloch, G. E., and Tangwongsan, K. Parallel approximation algorithms for facility-location problems. In *SPAA 2010: Proceedings of the 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures, Thira, Santorini, Greece, June 13-15, 2010* (2010), pp. 315–324.

[66] Boutsidis, C., Zouzias, A., Mahoney, M. W., and Drineas, P. Randomized dimensionality reduction for k-means clustering. *IEEE Transactions on Information Theory 61*, 2 (2015), 1045–1062.

[67] Bradley, P. S., Fayyad, U. M., and Reina, C. Scaling clustering algorithms to large databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City, New York, USA, August 27-31, 1998* (1998), pp. 9–15.

[68] Braverman, V., Lang, H., Levin, K., and Monemizadeh, M. Clustering on sliding windows in polylogarithmic space. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India* (2015), pp. 350–364.

[69] Braverman, V., Lang, H., Levin, K., and Monemizadeh, M. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016* (2016), pp. 1374–1390.

[70] Braverman, V., Meyerson, A., Ostrovsky, R., Roytman, A., Shindler, M., and Tagiku, B. Streaming k-means on well-clusterable data. In *SODA* (2011), pp. 26–40.

[71] Braverman, V., and Ostrovsky, R. Smooth histograms for sliding windows. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings* (2007), pp. 283–293.

[72] Braverman, V., Ostrovsky, R., and Vilenchik, D. How hard is counting triangles in the streaming model? In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I* (2013), pp. 244–254.

[73] Broder, A. Z. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997* (Washington, DC, USA, 1997), SEQUENCES '97, IEEE Computer Society, pp. 21–.

[74] Broder, A. Z. Identifying and filtering near-duplicate documents. In *Combinatorial Pattern Matching, 11th Annual Symposium, CPM 2000, Montreal, Canada, June 21-23, 2000, Proceedings* (2000), pp. 1–10.

[75] Broder, A. Z., Charikar, M., Frieze, A. M., and Mitzenmacher, M. Min-wise independent permutations. *J. Comput. Syst. Sci. 60*, 3 (2000), 630–659.

[76] Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G. Syntactic clustering of the web. *Computer Networks 29*, 8-13 (1997), 1157–1166.

[77] Brubaker, S. C., and Vempala, S. Isotropic PCA and affine-invariant clustering. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA* (2008), pp. 551–560.

[78] Buriol, L. S., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., and Sohler, C. Counting triangles in data streams. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)* (2006), pp. 253–262.

[79] Bury, M., and Schwiegelshohn, C. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings* (2015), pp. 263–274.

[80] Bury, M., and Schwiegelshohn, C. On finding the Jaccard center. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2016, Warsaw, Poland* (2017).

[81] Byrka, J., Pensyl, T., Rybicki, B., Srinivasan, A., and Trinh, K. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015* (2015), pp. 737–756.

[82] Cai, L., and Juedes, D. W. On the existence of subexponential parameterized algorithms. *J. Comput. Syst. Sci. 67*, 4 (2003), 789–807.

[83] Cesati, M., and Trevisan, L. On the efficiency of polynomial time approximation schemes. *Inf. Process. Lett. 64*, 4 (1997), 165–171.

[84] Chakrabarti, A., and Wirth, A. Incidence geometries and the pass complexity of semi-streaming set cover. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM*

Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016 (2016), pp. 1365–1373.

[85] Chan, T. M. Approximating the diameter, width, smallest enclosing cylinder, and minimum-width annulus. *Int. J. Comput. Geometry Appl. 12*, 1-2 (2002), 67–85.

[86] Chan, T. M. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. 35*, 1-2 (2006), 20–35.

[87] Chan, T. M. Dynamic streaming algorithms for epsilon-kernels. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA* (2016), pp. 27:1–27:11.

[88] Chan, T. M., and Pathak, V. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Comput. Geom. 47*, 2 (2014), 240–247.

[89] Chan, T. M., and Sadjad, B. S. Geometric optimization problems over sliding windows. *Int. J. Comput. Geometry Appl. 16*, 2-3 (2006), 145–158.

[90] Charikar, M. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada* (2002), pp. 380–388.

[91] Charikar, M., Chekuri, C., Feder, T., and Motwani, R. Incremental clustering and dynamic information retrieval. *SIAM J. Comput. 33*, 6 (2004), 1417–1440.

[92] Charikar, M., and Guha, S. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput. 34*, 4 (2005), 803–824.

[93] Charikar, M., O'Callaghan, L., and Panigrahy, R. Better streaming algorithms for clustering problems. In *STOC* (2003), pp. 30–39.

[94] Chazelle, B. *The discrepancy method - randomness and complexity.* Cambridge University Press, 2001.

[95] Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput. 39*, 3 (2009), 923–947.

[96] Chierichetti, F., and Kumar, R. LSH-preserving functions and their applications. *J. ACM 62*, 5 (2015), 33.

[97] Chierichetti, F., Kumar, R., Lattanzi, S., Mitzenmacher, M., Panconesi, A., and Raghavan, P. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009* (2009), pp. 219–228.

[98] Chierichetti, F., Kumar, R., Pandey, S., and Vassilvitskii, S. Finding the Jaccard median. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010* (2010), pp. 293–311.

[99] Chiplunkar, A., Tirodkar, S., and Vishwanathan, S. On randomized algorithms for matching in the online preemptive model. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings* (2015), pp. 325–336.

[100] Chitnis, R., Cormode, G., Esfandiari, H., Hajiaghayi, M., McGregor, A., Monemizadeh, M., and Vorotnikova, S. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016* (2016), pp. 1326–1344.

[101] Clarkson, K. L. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms 6*, 4 (2010).

[102] Clarkson, K. L., Hazan, E., and Woodruff, D. P. Sublinear optimization for machine learning. *J. ACM 59*, 5 (2012), 23.

[103] Clarkson, K. L., and Woodruff, D. P. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)* (2009), pp. 205–214.

[104] Clarkson, K. L., and Woodruff, D. P. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013* (2013), pp. 81–90.

[105] Clarkson, K. L., and Woodruff, D. P. Input sparsity and hardness for robust subspace approximation. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015* (2015), pp. 310–329.

[106] Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J. D., and Yang, C. Finding interesting associations without support pruning. *IEEE Trans. Knowl. Data Eng. 13*, 1 (2001), 64–78.

[107] Cohen, E., and Kaplan, H. Bottom-k sketches: better and more efficient estimation of aggregates. In *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2007, San Diego, California, USA, June 12-16, 2007* (2007), pp. 353–354.

[108] Cohen, E., and Kaplan, H. Summarizing data using bottom-k sketches. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007* (2007), pp. 225–234.

[109] Cohen, M. B., Elder, S., Musco, C., Musco, C., and Persu, M. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015* (2015), pp. 163–172.

[110] Cohen, M. B., Lee, Y. T., Miller, G. L., Pachocki, J., and Sidford, A. Geometric median in nearly linear time. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016* (2016), pp. 9–21.

[111] Cohen-Addad, V., Klein, P. N., and Mathieu, C. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA* (2016), pp. 353–364.

[112] Cohen-Addad, V., and Mathieu, C. Effectiveness of local search for geometric optimization. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands* (2015), pp. 329–343.

[113] Cohen-Addad, V., and Schwiegelshohn, C. On the local structure of stable clustering instances. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017), October 15-17, 2017, Berkeley, California, USA* (2017).

[114] Cohen-Addad, V., Schwiegelshohn, C., and Sohler, C. Diameter and k-center in sliding windows. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy* (2016), pp. 19:1–19:12.

[115] Cohen-Steiner, D., Alliez, P., and Desbrun, M. Variational shape approximation. *ACM Trans. Graph. 23*, 3 (2004), 905–914.

[116] Coja-Oghlan, A. Graph partitioning via adaptive spectral techniques. *Combinatorics, Probability & Computing 19*, 2 (2010), 227–284.

[117] Crouch, M., McGregor, A., and Stubbs, D. Dynamic graphs in the sliding-window model. In *Proceedings of the 21st Annual European Symposium (ESA)* (2013), pp. 337–348.

[118] Crouch, M., and Stubbs, D. Improved streaming algorithms for weighted matching, via unweighted matching. In *Proceedings of the 18th Workshop on Approximation,*

Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM) (2014), pp. 96–104.

[119] Czygrinow, A., Hanckowiak, M., and Szymanska, E. Fast distributed approximation algorithm for the maximum matching problem in bounded arboricity graphs. In *Proceedings of the 20th International Symposium on Symbolic and Algebraic Computation (ISSAC)* (2009), pp. 668–678.

[120] Das, A., Datar, M., Garg, A., and Rajaram, S. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007* (2007), pp. 271–280.

[121] Dasgupta, A., Hopcroft, J. E., Kannan, R., and Mitra, P. P. Spectral clustering with limited independence. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007* (2007), pp. 1036–1045.

[122] Dasgupta, S. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA* (1999), pp. 634–644.

[123] Dasgupta, S., and Schulman, L. J. A probabilistic analysis of EM for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research 8* (2007), 203–226.

[124] Datar, M., Gionis, A., Indyk, P., and Motwani, R. Maintaining stream statistics over sliding windows. *SIAM J. Comput. 31*, 6 (2002), 1794–1813.

[125] Datar, M., and Muthukrishnan, S. Estimating rarity and similarity over data stream windows. In *Algorithms - ESA 2002, 10th Annual European Symposium, Rome, Italy, September 17-21, 2002, Proceedings* (2002), pp. 323–334.

[126] de la Higuera, C., and Casacuberta, F. Topology of strings: Median string is NP-complete. *Theor. Comput. Sci. 230*, 1-2 (2000), 39–48.

[127] Demaine, E. D., Indyk, P., Mahabadi, S., and Vakilian, A. On streaming and communication complexity of the set cover problem. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings* (2014), pp. 484–498.

[128] Deshpande, A., and Vempala, S. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for*

Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings (2006), pp. 292–303.

[129] Dhillon, I. S., Guan, Y., and Kogan, J. Iterative clustering of high dimensional text data augmented by local search. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan* (2002), pp. 131–138.

[130] Dietzfelbinger, M., Hagerup, T., Katajainen, J., and Penttonen, M. A reliable randomized algorithm for the closest-pair problem. *J. Algorithms 25*, 1 (1997), 19–51.

[131] Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. Clustering large graphs via the singular value decomposition. *Machine Learning 56*, 1-3 (2004), 9–33.

[132] Duan, R., Pettie, S., and Su, H. Scaling algorithms for weighted matching in general graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19* (2017), pp. 781–800.

[133] Edmonds, J. Paths, trees, and flowers. *Canadian Journal of Mathematics 17* (1965), 449–467.

[134] Elkin, M. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Transactions on Algorithms 7*, 2 (2011), 20.

[135] Emek, Y., Halldórsson, M. M., and Rosén, A. Space-constrained interval selection. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I* (2012), pp. 302–313.

[136] Emek, Y., and Rosén, A. Semi-streaming set cover - (extended abstract). In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I* (2014), pp. 453–464.

[137] Epstein, L., Levin, A., Mestre, J., and Segev, D. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics 25*, 3 (2011), 1251–1265.

[138] Epstein, L., Levin, A., Segev, D., and Weimann, O. Improved bounds for online preemptive matching. In *Proceedings of the 30th Annual Symposium on Theoretical Aspects of Computer Science (STACS)* (2013), pp. 389–399.

[139] Esfandiari, H., Hajiaghayi, M. T., Liaghat, V., Monemizadeh, M., and Onak, K. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2015), pp. 1217–1233.

[140] Feder, T., and Greene, D. H. Optimal algorithms for approximate clustering. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA* (1988), pp. 434–444.

[141] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., and Zhang, J. On graph problems in a semi-streaming model. *Theoretical Computer Science 348*, 2-3 (2005), 207–216.

[142] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., and Zhang, J. Graph distances in the data-stream model. *SIAM Journal on Computing 38*, 5 (2008), 1709–1727.

[143] Feigenbaum, J., Kannan, S., and Zhang, J. Computing diameter in the streaming and sliding-window models. *Algorithmica 41*, 1 (2004), 25–41.

[144] Feigenblat, G., Porat, E., and Shiftan, A. Exponential time improvement for min-wise based algorithms. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011* (2011), pp. 57–66.

[145] Feldman, D., and Langberg, M. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011* (2011), pp. 569–578.

[146] Feldman, D., Monemizadeh, M., and Sohler, C. A PTAS for k-means clustering based on weak coresets. In *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007* (2007), pp. 11–18.

[147] Feldman, D., Schmidt, M., and Sohler, C. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013* (2013), pp. 1434–1453.

[148] Fellows, M. R., Gramm, J., and Niedermeier, R. On the parameterized intractability of CLOSEST substringsize and related problems. In *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings* (2002), pp. 262–273.

[149] Fichtenberger, H., Gillé, M., Schmidt, M., Schwiegelshohn, C., and Sohler, C. BICO: BIRCH meets coresets for k-means clustering. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings* (2013), pp. 481–492.

[150] Frahling, G., Indyk, P., and Sohler, C. Sampling in dynamic data streams and applications. *International Journal of Computational Geometry and Applications 18*, 1/2 (2008), 3–28.

[151] Frahling, G., and Sohler, C. Coresets in dynamic geometric data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)* (2005), pp. 209–217.

[152] Frahling, G., and Sohler, C. A fast k-means implementation using coresets. *Int. J. Comput. Geometry Appl. 18*, 6 (2008), 605–625.

[153] Frances, M., and Litman, A. On covering problems of codes. *Theory Comput. Syst. 30*, 2 (1997), 113–119.

[154] Friggstad, Z., Rezapour, M., and Salavatipour, M. R. Local search yields a PTAS for k-means in doubling metrics. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA* (2016), pp. 365–374.

[155] Friggstad, Z., and Zhang, Y. Tight analysis of a multiple-swap heurstic for budgeted red-blue median. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy* (2016), pp. 75:1–75:13.

[156] Gall, F. L. Powers of tensors and fast matrix multiplication. In *Proceedings of the 25th International Symposium on Symbolic and Algebraic Computation (ISSAC)* (2014), pp. 296–303.

[157] Garey, M. R., and Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[158] Gasieniec, L., Jansson, J., and Lingas, A. Efficient approximation algorithms for the Hamming center problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland.* (1999), pp. 905–906.

[159] Gavinsky, D., Kempe, J., Kerenidis, I., Raz, R., and de Wolf, R. Exponential separation for one-way quantum communication complexity, with applications to cryptography. *SIAM Journal on Computing 38*, 5 (2008), 1695–1708.

[160] Geelen, J. F. An algebraic matching algorithm. *Combinatorica 20*, 1 (2000), 61–70.

[161] Ghosh, D. Neighborhood search heuristics for the uncapacitated facility location problem. *European Journal of Operational Research 150*, 1 (2003), 150 – 162.

[162] Gionis, A., Indyk, P., and Motwani, R. Similarity search in high dimensions via hashing. In *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK* (1999), pp. 518–529.

[163] Goel, A., Kapralov, M., and Khanna, S. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2012), pp. 468–485.

[164] Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci. 38* (1985), 293–306.

[165] Gower, J. C., and Legendre, P. Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification 3*, 1 (1986), 5–48.

[166] Gramm, J., Niedermeier, R., and Rossmanith, P. Fixed-parameter algorithms for CLOSEST STRING and related problems. *Algorithmica 37*, 1 (2003), 25–42.

[167] Grigorescu, E., Monemizadeh, M., and Zhou, S. Estimating weighted matchings in o(n) space. *CoRR abs/1604.07467* (2016).

[168] Grigorescu, E., Monemizadeh, M., and Zhou, S. Streaming weighted matchings: Optimal meets greedy. *CoRR abs/1608.01487* (2016).

[169] Grötschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*, vol. 2 of *Algorithms and Combinatorics*. Springer, 1988.

[170] Guha, S. Tight results for clustering and summarizing data streams. In *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings* (2009), pp. 268–275.

[171] Guha, S., and Khuller, S. Greedy strikes back: Improved facility location algorithms. *J. Algorithms 31*, 1 (1999), 228–248.

[172] Guha, S., McGregor, A., and Tench, D. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015* (2015), pp. 241–247.

[173] Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. Clustering data streams: Theory and practice. *IEEE Trans. Knowl. Data Eng. 15*, 3 (2003), 515–528.

[174] Guha, S., Mishra, N., Motwani, R., and O'Callaghan, L. Clustering data streams. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA* (2000), pp. 359–366.

[175] Guha, S., Rastogi, R., and Shim, K. ROCK: A robust clustering algorithm for categorical attributes. *Inf. Syst. 25*, 5 (2000), 345–366.

[176] Guha, S., Rastogi, R., and Shim, K. Cure: An efficient clustering algorithm for large databases. *Inf. Syst. 26*, 1 (2001), 35–58.

[177] Gupta, A., and Tangwongsan, K. Simpler analyses of local search algorithms for facility location. *CoRR abs/0809.2554* (2008).

[178] Guruswami, V., and Indyk, P. Embeddings and non-approximability of geometric problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.* (2003), pp. 537–538.

[179] Guruswami, V., and Onak, K. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity (CCC)* (2013), pp. 287–298.

[180] Hall, P. On representatives of subsets. *Journal of the London Mathematical Society 10* (1935), 26–30.

[181] Halldórsson, B. V., Halldórsson, M. M., Losievskaja, E., and Szegedy, M. Streaming algorithms for independent sets. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)* (2010), pp. 641–652.

[182] Hansen, P., and Mladenovic, N. J-m$_{\text{eans}}$: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition 34*, 2 (2001), 405–413.

[183] Har-Peled, S., Indyk, P., Mahabadi, S., and Vakilian, A. Towards tight bounds for the streaming set cover problem. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016* (2016), pp. 371–383.

[184] Har-Peled, S., and Kushal, A. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry 37*, 1 (2007), 3–19.

[185] Har-Peled, S., and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004* (2004), pp. 291–300.

[186] Harvey, N. J. A. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput. 39*, 2 (2009), 679–702.

[187] Henzinger, M. R. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006* (2006), pp. 284–291.

[188] Henzinger, M. R., Raghavan, P., and Rajagopalan, S. Computing on data streams. In *External Memory Algorithms: DIMACS Workshop External Memory and Visualization* (1999), vol. 50, American Mathematical Society, pp. 107–118.

[189] Hochbaum, D. S., and Shmoys, D. B. A unified approach to approximation algorithms for bottleneck problems. *J. ACM 33*, 3 (1986), 533–550.

[190] Hopcroft, J. E., and Karp, R. M. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing 2*, 4 (1973), 225–231.

[191] Huang, Z., Radunović, B., Vojnović, M., and Zhang, Q. Communication complexity of approximate matching in distributed graphs. In *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS)* (2015).

[192] Impagliazzo, R., Paturi, R., and Zane, F. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci. 63*, 4 (2001), 512–530.

[193] Indyk, P. A small approximately min-wise independent family of hash functions. *J. Algorithms 38*, 1 (2001), 84–90.

[194] Indyk, P. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.* (2003), pp. 539–545.

[195] Indyk, P. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)* (2004), pp. 373–380.

[196] Indyk, P., and Motwani, R. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998* (1998), pp. 604–613.

[197] Indyk, P., and Price, E. K-median clustering, model-based compressive sensing, and sparse recovery for earth mover distance. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011* (2011), pp. 627–636.

[198] Jain, K., Mahdian, M., and Saberi, A. A new greedy approach for facility location problems. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada* (2002), pp. 731–740.

[199] Jain, K., and Vazirani, V. V. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM 48*, 2 (2001), 274–296.

[200] Jaiswal, R., and Garg, N. Analysis of k-means++ for separable data. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings* (2012), pp. 591–602.

[201] Janssens, S. *Bell inequalities in cardinality-based similarity measurement.* PhD thesis, Ghent University, 2006.

[202] Johnson, W. B., and Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics 26*, 189-206 (1984), 1–1.

[203] Jowhari, H., and Ghodsi, M. New streaming algorithms for counting triangles in graphs. In *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON)* (2005), pp. 710–716.

[204] Jowhari, H., Saglam, M., and Tardos, G. Tight bounds for l_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)* (2011), pp. 49–58.

[205] Kane, D. M., Nelson, J., and Woodruff, D. P. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA* (2010), pp. 41–52.

[206] Kannan, R., Salmasian, H., and Vempala, S. The spectral method for general mixture models. *SIAM J. Comput. 38*, 3 (2008), 1141–1156.

[207] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 7 (2002), 881–892.

[208] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. A local search approximation algorithm for k-means clustering. *Comput. Geom. 28*, 2-3 (2004), 89–112.

[209] Kapralov, M. Better bounds for matchings in the streaming model. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2013), pp. 1679–1697.

[210] Kapralov, M., Khanna, S., and Sudan, M. Approximating matching size from random streams. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2014), pp. 734–751.

[211] Kapralov, M., Khanna, S., and Sudan, M. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2015), pp. 1263–1282.

[212] Kapralov, M., Khanna, S., Sudan, M., and Velingker, A. $(1 + \omega(1))$-approximation to MAX-CUT requires linear space. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19* (2017), pp. 1703–1722.

[213] Kapralov, M., Lee, Y. T., Musco, C., Musco, C., and Sidford, A. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014* (2014), pp. 561–570.

[214] Kapralov, M., and Woodruff, D. P. Spanners and sparsifiers in dynamic streams. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014* (2014), pp. 272–281.

[215] Kelner, J. A., and Levin, A. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems 53*, 2 (2013), 243–262.

[216] Kerber, M., and Raghvendra, S. Approximation and streaming algorithms for projective clustering via random projections. In *Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG 2015, Kingston, Ontario, Canada, August 10-12, 2015* (2015).

[217] Kim, S., and Ahn, H. An improved data stream algorithm for clustering. *Comput. Geom. 48*, 9 (2015), 635–645.

[218] Kolliopoulos, S. G., and Rao, S. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM J. Comput. 37*, 3 (June 2007), 757–782.

[219] Konrad, C. Maximum matching in turnstile streams. *ArXiv e-prints* (2015), 1505.01460.

[220] Konrad, C., Magniez, F., and Mathieu, C. Maximum matching in semi-streaming with few passes. In *Proceedings of the 16th Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)* (2012), pp. 231–242.

[221] Korupolu, M. R., Plaxton, C. G., and Rajaraman, R. Analysis of a local search heuristic for facility location problems. *J. Algorithms 37*, 1 (2000), 146–188.

[222] Kumar, A., and Kannan, R. Clustering with spectral norm and the k-means algorithm. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA* (2010), pp. 299–308.

[223] Kumar, A., Sabharwal, Y., and Sen, S. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM 57*, 2 (2010).

[224] Kumar, P., Mitchell, J. S. B., and Yildirim, E. A. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM Journal of Experimental Algorithmics 8* (2003).

[225] Lanctôt, J. K., Li, M., Ma, B., Wang, S., and Zhang, L. Distinguishing string selection problems. *Inf. Comput. 185*, 1 (2003), 41–55.

[226] Langberg, M., and Schulman, L. J. Universal epsilon-approximators for integrals. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010* (2010), pp. 598–607.

[227] Lattanzi, S., Moseley, B., Suri, S., and Vassilvitskii, S. Filtering: a method for solving graph problems in mapreduce. In *Proceedings of the 23rd annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)* (2011), pp. 85–94.

[228] Li, M., Ma, B., and Wang, L. On the closest string and substring problems. *J. ACM 49*, 2 (2002), 157–171.

[229] Li, P., and König, A. C. Theory and applications of b-bit minwise hashing. *Commun. ACM 54*, 8 (2011), 101–109.

[230] Li, S., and Svensson, O. Approximating k-median via pseudo-approximation. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013* (2013), pp. 901–910.

[231] Li, Y., Nguyen, H. L., and Woodruff, D. P. On sketching matrix norms and the top singular vector. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2014), pp. 1562–1581.

[232] Li, Y., Nguyen, H. L., and Woodruff, D. P. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)* (2014), pp. 174–183.

[233] Li, Y., and Woodruff, D. P. On approximating functions of the singular values in a stream. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016* (2016), pp. 726–739.

[234] Lovász, L. On determinants, matchings, and random algorithms. In *Proceedings of the 2nd Conference on Fundamentals of Computation Theory (FCT)* (1979), pp. 565–574.

[235] Lovász, L., and Plummer, M. *Matching theory*, vol. 367. American Mathematical Soc., 2009.

[236] MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* (Berkeley, Calif., 1967), University of California Press, pp. 281–297.

[237] Madry, A. Navigating central path with electrical flows: From flows to matchings, and back. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA* (2013), pp. 253–262.

[238] Makarychev, K., and Makarychev, Y. Metric perturbation resilience. *CoRR abs/1607.06442* (2016).

[239] Marx, D. Closest substring problems with small distances. *SIAM J. Comput. 38*, 4 (2008), 1382–1410.

[240] Matousek, J. On approximate geometric k-clustering. *Discrete & Computational Geometry 24*, 1 (2000), 61–84.

[241] Matousek, J., Sharir, M., and Welzl, E. A subexponential bound for linear programming. *Algorithmica 16*, 4/5 (1996), 498–516.

[242] McCutchen, R. M., and Khuller, S. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings* (2008), pp. 165–178.

[243] McGregor, A. Finding graph matchings in data streams. In *Proceedings of the 9th Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)* (2005), pp. 170–181.

[244] McGregor, A. Graph stream algorithms: a survey. *SIGMOD Record 43*, 1 (2014), 9–20.

[245] McGregor, A., Tench, D., Vorotnikova, S., and Vu, H. T. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II* (2015), pp. 472–482.

[246] McGregor, A., and Vorotnikova, S. Planar matching in streams revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France* (2016), pp. 17:1–17:12.

[247] McGregor, A., Vorotnikova, S., and Vu, H. T. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016* (2016), pp. 401–411.

[248] McSherry, F. Spectral partitioning of random graphs. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA* (2001), pp. 529–537.

[249] Mettu, R. R., and Plaxton, C. G. The online median problem. *SIAM J. Comput. 32*, 3 (2003), 816–832.

[250] Micali, S., and Vazirani, V. V. An $\mathcal{O}(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (1980), pp. 17–27.

[251] Mislove, A., Marcon, M., Gummadi, P. K., Druschel, P., and Bhattacharjee, B. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM Internet Measurement Conference, IMC 2007, San Diego, California, USA, October 24-26, 2007* (2007), pp. 29–42.

[252] Mitzenmacher, M., and Upfal, E. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[253] Mucha, M., and Sankowski, P. Maximum matchings via Gaussian elimination. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (2004), pp. 248–255.

[254] Muthukrishnan, S. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science 1*, 2 (2005).

[255] Naish, L., Lee, H. J., and Ramamohanarao, K. A model for spectra-based software diagnosis. *ACM Trans. Softw. Eng. Methodol. 20*, 3 (2011), 11.

[256] Nicolas, F., and Rivals, E. Hardness results for the center and median string problems under the weighted and unweighted edit distances. *J. Discrete Algorithms 3*, 2-4 (2005), 390–415.

[257] Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. The effectiveness of Lloyd-type methods for the k-means problem. *J. ACM 59*, 6 (2012), 28.

[258] Pagh, A., and Pagh, R. Uniform hashing in constant time and optimal space. *SIAM J. Comput. 38*, 1 (2008), 85–96.

[259] Pagh, R., Stöckel, M., and Woodruff, D. P. Is min-wise hashing optimal for summarizing set intersection? In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014* (2014), pp. 109–120.

[260] Paz, A., and Schwartzman, G. A $(2 + \varepsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19* (2017), pp. 2153–2161.

[261] Phillips, J. M., Verbin, E., and Zhang, Q. Lower bounds for number-in-hand multiparty communication complexity, made easy. *SIAM J. Comput. 45*, 1 (2016), 174–196.

[262] Pisier, G. *The volume of convex bodies and Banach space geometry.* Cambridge Tracts in Mathematics. 94, 1999.

[263] Popov, V. Y. Multiple genome rearrangement by swaps and by element duplications. *Theor. Comput. Sci. 385*, 1-3 (2007), 115–126.

[264] Rabin, M. O., and Vazirani, V. V. Maximum matchings in general graphs through randomization. *J. Algorithms 10*, 4 (1989), 557–567.

[265] Rieck, K., Trinius, P., Willems, C., and Holz, T. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security 19*, 4 (2011), 639–668.

[266] Sankowski, P. Processor efficient parallel matching. *Theory Comput. Syst. 42*, 1 (2008), 73–90.

[267] Sarlós, T. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)* (2006), pp. 143–152.

[268] Schulman, L. J. Clustering for edge-cost minimization (extended abstract). In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA* (2000), pp. 547–555.

[269] Shindler, M., Wong, A., and Meyerson, A. Fast and accurate k-means for large datasets. In *NIPS* (2011), pp. 2375–2383.

[270] Späth, H. The minisum location problem for the Jaccard metric. *Operations-Research-Spektrum 3*, 2 (1981), 91–94.

[271] Spielman, D. A., and Teng, S.-H. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM 51*, 3 (2004), 385–463.

[272] Sun, M. Solving the uncapacitated facility location problem using tabu search. *Computers & OR 33* (2006), 2563–2589.

[273] Sundaram, N., Turmukhametova, A., Satish, N., Mostak, T., Indyk, P., Madden, S., and Dubey, P. Streaming similarity search over one billion tweets using parallel locality-sensitive hashing. *PVLDB 6*, 14 (2013), 1930–1941.

[274] Thorup, M. Bottom-k and priority sampling, set similarity and subset sums with minimal independence. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013* (2013), pp. 371–380.

[275] Thorup, M., and Zhang, Y. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput. 41*, 2 (2012), 293–331.

[276] Trefethen, L. N., and Bau, D. *Numerical Linear Algebra*. SIAM, 1997.

[277] Tutte, W. T. The factorization of linear graphs. *Journal of the London Mathematical Society 22* (1947), 107–111.

[278] Tuzun, D., and Burke, L. I. A two-phase tabu search approach to the location routing problem. *European journal of operational research 116*, 1 (1999), 87–99.

[279] Uehara, R., and Chen, Z.-Z. Parallel approximation algorithms for maximum weighted matching in general graphs. *Information Processing Letters 76*, 1-2 (2000), 13–17.

[280] Varadaraja, A. B. Buyback problem - approximate matroid intersection with cancellation costs. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I* (2011), pp. 379–390.

[281] Vempala, S., and Wang, G. A spectral algorithm for learning mixture models. *J. Comput. Syst. Sci. 68*, 4 (2004), 841–860.

[282] Verbin, E., and Yu, W. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the 22th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2011), pp. 11–25.

[283] Watson, G. A. An algorithm for the single facility location problem using the Jaccard metric. *SIAM Journal on Scientific and Statistical Computing 4*, 4 (1983), 748–756.

[284] Yildirim, E. A. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization 19*, 3 (2008), 1368–1391.

[285] Zadeh, R. B., and Goel, A. Dimension independent similarity computation. *Journal of Machine Learning Research 14*, 1 (2013), 1605–1626.

[286] Zarrabi-Zadeh, H. Core-preserving algorithms. In *Proceedings of the 20th Annual Canadian Conference on Computational Geometry, Montréal, Canada, August 13-15, 2008* (2008).

[287] Zarrabi-Zadeh, H. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica 60*, 1 (2011), 46–59.

[288] Zelke, M. Weighted matching in the semi-streaming model. *Algorithmica 62*, 1-2 (2012), 1–20.

[289] Zhang, T., Ramakrishnan, R., and Livny, M. BIRCH: A new data clustering algorithm and its applications. *Data Min. Knowl. Discov. 1*, 2 (1997), 141–182.