Summer 6-6-2021

Counting and Sampling Small Structures in Graph and Hypergraph Data Streams

Themistoklis Haris
Themistoklis.Haris.21@Dartmouth.edu

Counting and Sampling Small Structures in Graph and Hypergraph Data Streams

Themistoklis Haris

Advisor: Amit Chakrabarti

May 31, 2021

An Undergraduate Thesis

Dartmouth College, Department of Computer Science

Abstract

In this thesis, we explore the problem of approximating the number of elementary substructures called *simplices* in large k-uniform *hypergraphs*. The hypergraphs are assumed to be too large to be stored in memory, so we adopt a data stream model, where the hypergraph is defined by a sequence of hyperedges.

First we propose an algorithm that (ε, δ)-estimates the number of simplices using $\widetilde{O}\left(\frac{m^{1+\frac{1}{k}}}{T}\right)$ bits of space. In addition, we prove that no constant-pass streaming algorithm can (ε, δ)-approximate the number of simplices using less than $O\left(\frac{m^{1+\frac{1}{k}}}{T}\right)$ bits of space. Thus we resolve the space complexity of the simplex counting problem by providing an algorithm that matches the lower bound.

Second, we examine the triangle counting question –a hypergraph where $k = 2$. We develop and analyze an almost optimal $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$ triangle-counting algorithm based on ideas introduced in [KMPT12]. The proposed algorithm is subsequently used to establish a method for uniformly sampling triangles in a graph stream using $\widetilde{O}\left(\frac{m^{3/2}}{T}\right)$ bits of space, which beats the state-of-the-art $\widetilde{O}\left(\frac{mn}{T}\right)$ algorithm given by [PTTW13].

Acknowledgements

I would not have been able to complete this thesis in its present form if it wasn't for the help of many people around me. I want to extend my sincerest gratitude to everyone who helped me make this thesis possible. Professor Amit, for his tireless support and incredibly helpful feedback as this work went on. Professors Deeparnab Chakrabarty and Hsien-Chih Chang for being in my thesis committee and providing me with a lot of valuable advice and mentorship. My mom, for encouraging me to be resilient in times when problems appear overwhelming. My dad, for patiently and happily listening to my thought process throughout various stages of this work. Linda, for supporting and encouraging me since the beginning and for reminding me to set daily thesis goals. Finally, all the friends and family that I have not mentioned: thank you all for caring so much about me.

Contents

Chapter 1

Introduction

Networks appear everywhere around us. From metabolic networks, which describe biological chemical reactions, to economic networks, which outline the interactions taking place within a closed market. An incredible amount of mathematical and empirical knowledge has been generated for the purpose of understanding the structure of networks, and yet there are still many mysteries that remain unsolved.

With the advent of the Digital Revolution, computers been used extensively in the analysis and generation of massive networks which facilitate the storage and exchange of information. However, as humanity dives deeper into the era of Big Data, even the fastest supercomputers in the world seem incapable of processing such large networks. Hence, a considerable amount of academic and industrial research today is focused on designing efficient and practical methods for network analysis.

This thesis explores a very specific but also recurring problem in network analysis - the problem of detecting and counting small patterns in networks. Network patterns are specific substructures which repeat themselves inside the network. For example, the simplest pattern in a network is the *triangle*, which describes three interconnected agents. Other examples of particularly interesting and well-studied patterns in networks are *cycles* and *cliques*.

The study of network patterns finds numerous applications in real-world situations. In a 2009 study, Christakis and Fowler [CF10] analyzed the spread of a flu epidemic at Harvard College, and proposed a simple, yet surprisingly effective method for identifying individuals that are central to the spread of the virus. They surveyed students at random and groups of their friends, and discovered that, on average, friend groups were infected by the virus much earlier than the general population, indicating that they played a more central role in the development of that epidemic. This phenomenon is an example of the *friendship paradox* in human social networks; that one's friends typically have more friends than they do.

The aforementioned study also surveyed a number of network metrics and their association to early contagion. One of those metrics was the *transitivity* metric, which represents the probability that two of one's friends are also friends. Transitivity in a network is equal to the ratio between the number of triangles and the number of length-two paths. The authors in [CF10] found that transitivity is negatively associated to early contagion. Intuitively, an individual that participates in few triangles is a part of many independent communities, and so they have a higher chance of contracting and spreading the virus.

Indeed, fast and efficient pattern analysis is an integral part of problem solving techniques

across many different fields. Triangles have been extensively used as a network topological feature to detect product review spamming in websites such as Amazon or eBay [WCW+18, KMPT12]. In the field of Computer-Aided Design (CAD), pattern counting is an important subroutine to solving systems of geometric constraints [KMPT12, FH97].

But perhaps most prominently of all, the study of network patterns is featured in a lot of recent scientific work on social networks. Concepts like "the friend of my friend is my friend" or "the enemy of my friend is my enemy" give rise to theories of *balance* and *status* that explain many collective phenomena observed in society, in online social networks, and in the animal kingdom [KMPT12, LHK10]. Knowledge about the structure and properties of these social networks can have a profound effect on policy design, advertising, online privacy and security.

In the remainder of this introduction, an informal outline is provided for the main technical problem this thesis attempts to tackle. Furthermore, an exposition is given on the preceding work and research that motivated, inspired and guided this thesis. Finally, the main results of this thesis are accumulated at the end of the chapter for the convenience of the reader.

1.1 Problem Formulation

1.1.1 Streaming Algorithms

Information is typically communicated between computers or databases in a sequential fashion. Packets of information are encrypted by servers, sent through secure channels, and decrypted by clients. As the world of computing is taken over by the Big Data trend, datasets are observed to grow in size exponentially fast. Indeed, 18 zettabytes of information were created, stored and consumed overall in 2016, 41 in 2019 and (projected) 118 in 2023. Thus, there is an increasing demand for algorithms that operate with very low memory requirements.

The **streaming model** is a design model for algorithms whose input is a sequence of data, or a **stream**. The goal of a streaming algorithm is to infer a target statistic about its input stream, while avoiding to store the entire stream in memory. This model captures both the sequential nature of information exchange in the digital world and the necessity for memory efficiency in the age of Big Data. Overall, the streaming model is very well studied, with representative problems such as *frequency estimation*, *frequency-moment estimation* and *dimensionality reduction* enjoying elegant and highly efficient solutions.

Streaming algorithms for networks are very relevant to the efficient processing of huge network datasets. For example, social networks such as Facebook or Twitter, with billions of vertices and trillions of edges, are often mined using Machine Learning algorithms for the inference of social trends. These Machine Learning algorithms make extensive use of various features of the network, like its number of connected components, or its diameter. Due to the immense size of the datasets, any feasible algorithm for computing such network statistics must use as little memory as possible. Further, information about the network is usually provided as a data stream from some server, which makes the streaming model ideal as a context of studying network analysis.

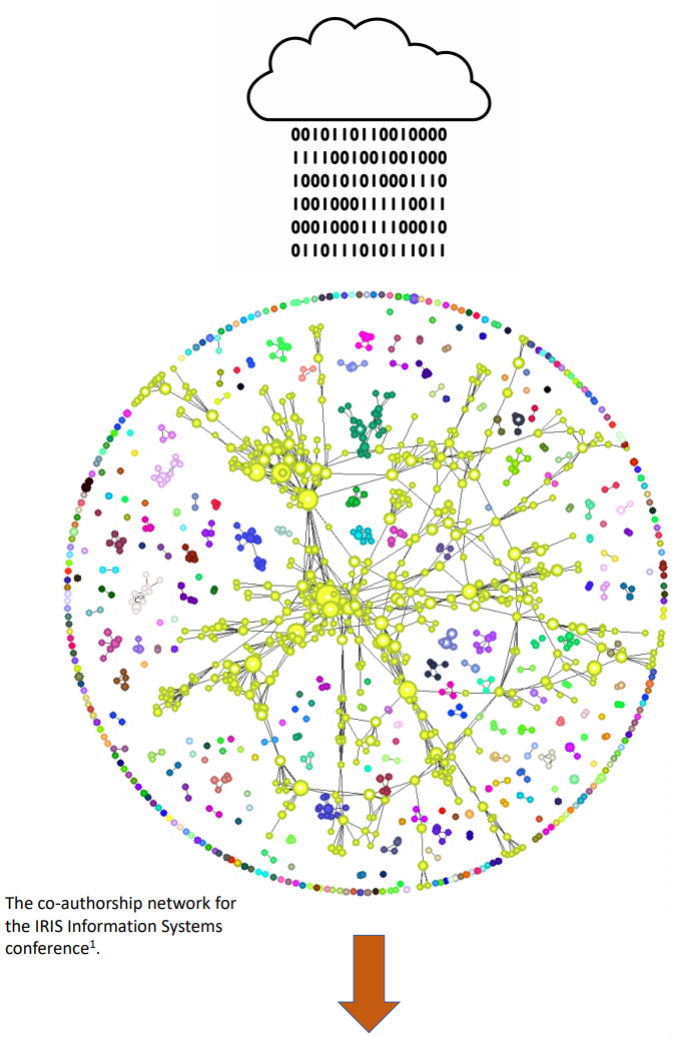

Figure 1.1: A standard data processing framework involving the data stream model. A cloud server provides a stream of information describing a network.

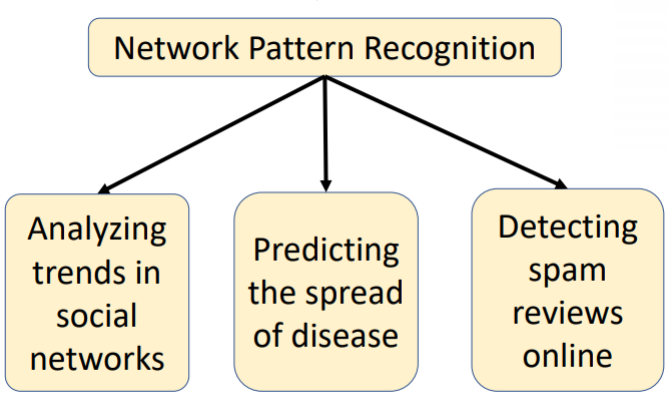

Figure 1.2: Some applications of network pattern counting

1.1.2 Hypergraphs and Generalized Networks

Typically, a network is mathematically abstracted into a **graph**. In a graph, each *edge* connects two *vertices*, representing a relationship/interaction between two entities in the network environment.

However, there are situations in which this two-sided relationship is not general enough of an abstraction. Consider, for example, a Computer Science conference [Bre13], in which multiple papers are presented. Each paper is the collective work of one or more computer scientists. To model the interactions in this network, it may not be enough to connect just two scientists when they participate in the same paper, because they may participate in many common projects with other, different scientists. To capture the network configuration fully, we have to connect all the scientists who are writing a paper together simultaneously.

As a result, the notion of a **hypergraph** or **set-system** is born. Before the 1990s, the term "set-system" appears more often in the mathematical literature, stemming from the viewpoint of edges as subsets of the network vertex universe. Hypergraphs consist of **hyperedges** that connect one or more vertices simultaneously, instead of strictly two. They have been the focus of considerable research in mathematics, although many of their structural properties have not yet been fully understood. Beyond theory, hypergraphs find numerous applications in Database Systems, Image Processing, Telecommunications, and Computational Chemistry, rendering them a powerful and valuable construct in many fields.

1.1.3 Main Problem Statement

Patterns in hypergraphs are more complicated and diverse than typical patterns in graphs. This thesis will be primarily focused on the problem of counting and detecting a specific set of patterns in **k-uniform hypergraphs**; that is hypergraphs whose hyperedges connect exactly k vertices.

The patterns of interest are known as *simplices*. In a k-uniform hypergraph, a **simplex** is a set of $(k+1)$ vertices in which every possible hyperedge exists. For example, if $k = 2$, a simplex is a triangle, and if $k = 3$, vertices $\{a, b, c, d\}$ are a simplex when $\{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}$ are all hyperedges of the hypergraph. For the case $k = 3$, a simplex could be thought of as a *tetrahedron* in space, where each face is a hyperedge.

The primary design problem this thesis will address can be summarized as follows:

Problem Statement: *Given a k-uniform hypergraph $\mathcal{H} = (V, \mathcal{E})$, design a streaming algorithm to calculate or estimate with high accuracy and confidence the number of simplices in $\mathcal{H}$, using as little memory as possible.*

In the effort to design the most memory-efficient algorithm possible, it is natural for one to wonder how efficient such an algorithm can be. This brings one to the following problem:

Problem Statement: *Let $\mathcal{H} = (V, \mathcal{E})$ be a k-uniform hypergraph with n vertices, m hyperedges and T simplices. Asymptotically, how many bits of memory must any algorithm use at least, to determine or estimate T in the worst case? The answer should be a function of n, m, k and potentially T.*

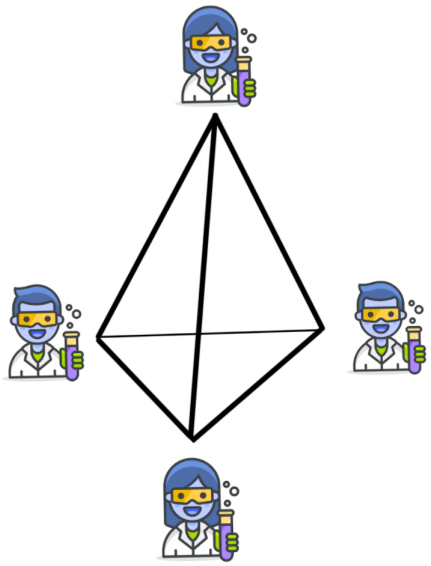

Figure 1.3: A coauthor network can be modeled as a hypergraph: the scientists who worked together are joined by a single connection. In a hypergraph, the type of pattern where every 3 nodes are connected to each other is called a simplex.

1.1.4 Other problems studied

A few other related problems are also considered throughout this thesis, but occupy considerably less space in the final exposition. Those are listed below:

1. **Triangle Counting in graphs:** The problem of counting triangle patterns in graphs is central in the field of streaming algorithms for networks. This thesis makes a small contribution to this big collective work by proposing a slightly different method for solving this important problem.

2. **Triangle Sampling in graphs:** Sampling a triangle uniformly from a graph stream is a problem which has apparently only been addressed directly by [PTTW13] in the literature. This thesis proposes a more efficient method to solve this problem.

1.2 Related work

Triangle counting has received a lot of attention in the streaming literature.

In field of streaming algorithms, the seminal paper by Bar-Yossef et al [BYKS02] initiated the study of this problem. The authors discovered a very clever reduction of the triangle counting problem to frequency-moment estimation, thus giving a $\widetilde{O}\left(\left(\frac{mn}{T}\right)^2\right)$-space[1] algorithm which uses only one pass over the input stream. The main idea behind their technique is to construct a virtual stream of vertex triples $\{u, v, w\}$ for every edge $\{u, v\}$ that arrives in the stream. Then, the triangle count of the graph can be found by solving a linear system of equations, in terms of the frequency moments of the virtual stream. The downside of this

[1] ε, δ factors will be hidden inside the $\widetilde{O}$ notation

algorithm is that it blows up the length of the stream - which is already very large - by a factor of n, which leads to a considerable memory overhead.

Buriol et al [BFL+06] produced a single-pass $\widetilde{O}\left(\frac{mn}{T}\right)$ streaming algorithm for counting triangles by using sampling techniques. Their work initiates a relatively long line of research in which algorithms attempt to count patterns in graph streams by sampling certain "characteristic" structures that are embedded in the pattern of interest. In [BFL+06], this structure is a random edge-vertex pair. The paradigm is very close to *importance sampling*, where the algorithms are designed to "narrow down" the universe of vertex triples in which the triangles are contained in. Indeed, this is what the main algorithms presented in this thesis will also attempt to do. The success and efficiency of an algorithm designed like this is intuitively determined by how "narrow" the universe is made to be. In the example of [BFL+06], edge-vertex pairs are not the strictest structure that could possibly label a triangle, and so there is still room for improvement to this algorithm.

Subsequent work resulted in more efficient algorithms for triangle counting through sampling. McGregor et al [MVV16] use a degree-ordering idea to label triangles with "ordered wedges" that can be sampled easily through ℓ_2-sampling and a degree oracle. This leads to a $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$ algorithm, which is very close to the theoretical optimum (more on lower bounds to follow). Chakrabarti and Bera [BC17] manage to remove the need for a degree oracle by first sampling the wedge and then checking if it is degree consistent. This yields an optimal $\widetilde{O}\left(\frac{m^{3/2}}{T}\right)$-space algorithm, but adds an additional pass over the input stream. Their algorithm also has the advantage that it can be generalized to counting constant size cliques or even general subgraphs.

Another line of attack to the triangle counting problem has been through sparsification techniques. In their highly influencial paper [TKMF09], Tsourakakis and Faloutsos propose "Doulion", an algorithm framework that first sparsifies a graph before counting its triangles. The authors prove that if each edge is kept with probability p and $p^3 = \widetilde{\Omega}\left(\frac{\Delta \log n}{T}\right)$, where Δ is the highest degree, then the triangle count of the sparsified graph is highly concentrated around T/p^3. This framework complements most triangle counting algorithms, streaming and non-streaming, and works very well in practice. The interested reader can refer to [Her20] for a survey on sparsification techniques in triangle counting.

The concept of "heavy" - "light" graph structures has also found useful applications in this problem, and this thesis will also use it on occasion. Kolountzakis et al [KMPT12] split the vertex set into heavy and light vertices based on their degree, which allows them to sample triples in a very efficient way. They outline a $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$-space algorithm for counting triangles in the streaming model, but assuming that the edges incident to a vertex come in order.

Cormode et al [CJ17] distinguish between heavy and light edges, based on how many triangles an edge participates in. Also seen in [TKMF09], the key insight is that heavy edges increase the variance of the triangle-count estimator. So, the algorithm tries to remove them and treat them separately. To this end, [CJ17] uses a heavy-light oracle, which is a data structure that allows an algorithm, with high probability, to predict the heaviness of an edge. This idea has also been used by McGregor and Vorotnikova [MV20, Vor20] to count 4-cycles in a graph.

Other work on triangle counting streaming algorithms includes working on dynamic

streams [SOK+20, BFKP16], degeneracy-based techniques [BS20], triangle sampling algorithms [PTTW13] and coloring-based algorithms [PT12].

Through the use of communication complexity, lower bounds for the triangle counting problem have also been discovered. Most of the related arguments are made by reducing from the disjointness communication problem. The optimal lower bounds, which also have matching algorithms up to (ε, δ) factors, are $\Omega\left(\frac{m^{3/2}}{T}\right)$ and $\Omega\left(\frac{m}{\sqrt{T}}\right)$ in the worst case. T here is a lower bound to the triangle-count of a graph. It is important to note that the most general lower bound is of the form $\Omega(n^2)$, even for an (ε, δ)-approximation. For detailed expositions on these arguments, please refer to [BC17, MVV16, CJ17, Vor20, BOV13].

For the problem of directed triangle and substructure counting (or *motif* counting), there has not been a lot of work in the streaming literature, with a few isolated exceptions like [BDGL08] and [SJHS15]. One possible reason for this lack of work on this problem might be that the algorithms for counting undirected patterns can easily be generalized to count directed motifs. Nevertheless, directed graph motifs are important in network analysis, so carefully outlined streaming algorithms for counting, detecting and enumerating directed patterns also have value in both academic and industrial settings.

Finally, hypergraph networks have been studied in the streaming literature, although nowhere near as extensively as triangles or other mainstream patterns. Guha and McGregor [GMT15] use the power of linear sketches to test for connectivity in hypergraph streams. Kallaugher et al [KKP18] study the sketching complexity of subhypergraph counting, giving upper and lower bounds to the general problem in terms of the vertex cover size of the queried subhypergraph.

Sun [Sun13] gives a general algorithm counting for hypergraph patterns with k edges in the turnstile streaming model using $\widetilde{O}\left(\frac{m^k}{T^2}\right)$-bits of space. This algorithm is based on complex-valued random variables and graph polynomials and has the advantage that it works on dynamic graph streams. However, at least for the purposes of the simplex-counting problem, this algorithm is suboptimal, and overly complicated to implement.

1.3 Main Results

The main contribution of this thesis is the resolution of the Simplex Counting problem. The main, overarching theorem is given below:

Theorem 1.3.1 (Main Theorem). *Let $\mathcal{H} = (G, \mathcal{E})$ be a k-uniform hypergraph with m hyperedges and S simplices. Let ε, δ be positive constants. There exists a 4-pass streaming algorithm that can (ε, δ)-approximate S using $\widetilde{O}\left(\frac{m^{1+\frac{1}{k}}}{S}\right)$ bits of space. Further, no streaming algorithm can produce such an estimate of S using $o\left(\frac{m^{1+\frac{1}{k}}}{S}\right)$ bits of space in the worst case.*

This theorem combines an upper bound and a lower bound result, so it will be shown in two separate chapters later on in this thesis. An interesting observation one can make right after seeing this result is that as $k \to \infty$, the memory required to solve the simplex counting problem converges to $\frac{m}{S}$. Intuitively, this can be explained by the fact that increasing k

while keeping m constant makes simplices far more restrictive hypergraph substructures. Therefore one is able to detect, reject and count them more easily. This idea will become formalized later on in the design of an algorithm to justify Theorem 1.3.1.

Another small caveat to notice with the previous theorem is the assumption that k is a constant. The algorithm designed later on hides factor of 2^k in its memory usage, but k is considered a constant, so it is left out. Therefore, we implicitly make the reasonable assumption that $k \ll m$ for the simplex-counting algorithm to work reasonably well in practice.

A secondary contribution of this thesis is an efficient triangle counting algorithm and uniform triangle sampling method.

Theorem 1.3.2. *Let $G = (V, E)$ be an undirected graph with m edges and T triangles. Algorithm 1, designed in Chapter 4, (ε, δ)-approximates T using $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$ bits of space in the worst case.*

Theorem 1.3.3. *Let $G = (V, E)$ be an undirected graph with m edges and T triangles. There is a 3-pass, $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$-space, streaming algorithm which can sample a triangle from G uniformly at random.*

The algorithm of Theorem 1.3.3 is, to our knowledge, one of the few algorithms in the streaming literature which solve the problem of uniform triangle sampling, and potentially the most efficient one to date to do so.

1.4 Thesis Outline

This thesis is organizes as follows:

1. **Chapter 2** will be devoted on presenting some of the prerequisite knowledge needed for reading and comprehending the material to follow. Also the notation used throughout this thesis is presented there.
2. **Chapter 3** will present some novel mathematical results which have been derived to assist in the analysis of the algorithms presented later on.
3. **Chapter 4** is focused on the triangle counting and sampling results stated above.
4. **Chapter 5** provides algorithms for simplex counting in hypergraphs.
5. **Chapter 6** uses communication theory techniques to show lower bounds to the simplex counting problem in hypergraphs
6. **Chapter 7** provides a few concluding remarks and future lines of related research

Chapter 2

Preliminaries

2.1 Graphs and Hypergraphs

Given a non-empty, finite set V and a non-empty subset $\mathcal{E}$ of 2^V, the pair $\mathcal{H} = (V, \mathcal{E})$ is called a **hypergraph**. The elements of V are called **vertices** and the subsets in $\mathcal{E}$ are called **hyperedges**. If $|E| = k \geq 1$ for all $E \in \mathcal{E}$, then $\mathcal{H}$ is said to be k**-uniform**.

For $X \subseteq \mathcal{E}$, let $\mathcal{V}(X)$ be the set of all vertices in V that belong to some hyperedge in X. For $S \subseteq V$, let $\mathcal{E}(S)$ be the set of hyperedges contained completely within S. We call $\mathcal{E}(S)$ the set of **induced** hyperedges of S.

Let $S \subseteq V$ be some subset of the vertex set, where $|S| < k$. Then the **degree** of S is defined to be the number of edges containing S. If $S = \emptyset$, then $\deg(S) = |\mathcal{E}|$ by default. We write:

$$\deg(S) := |\{e \in \mathcal{E} \mid S \subseteq e\}| \tag{2.1}$$

The **neighborhood** N_S of S is the set of vertices who share some edge with S:

$$N_S := \{v \in V \mid \exists e \in \mathcal{E} : S \cup \{v\} \subseteq e\} \tag{2.2}$$

It is important to note that in general, $|N_S| \neq \deg(S)$. If, for example, $S = \{u\}$, then every edge containing u adds at most $k - 1$ new neighbors, so we have that $|N_v| \leq (k-1)\deg(u)$.

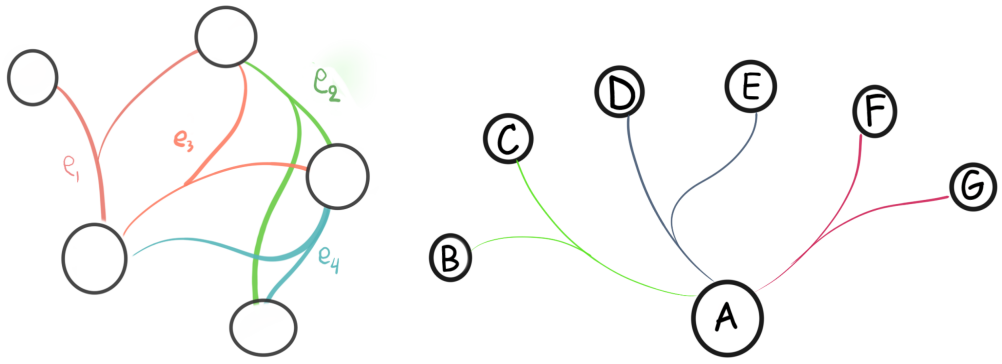

Figure 2.1: *Left*: A 3-uniform hypergraph, *Right*: Vertex A has degree 3 but $N_A = 6$.

For a subset $R \subseteq V$, the S-**codegree** of R is defined to be degree of the set $S \cup R$:

$$\deg(S, R) = \deg(S \cup R) \tag{2.3}$$

We can also define **the S-hypergraph of $\mathcal{H}$, $\mathbf{G_S} = (\mathbf{V} \setminus \mathbf{S}, \mathcal{E_S})$** to be the $(k - |S|)$-uniform hypergraph where

$$e = \{u_1, ..., u_{k-|S|}\} \in \mathcal{E}_S \text{ iff } e \cup S \in \mathcal{E} \tag{2.4}$$

It is straightforward to see that the S-codegree of a subset $R \subseteq V$ is equal to its degree in G_S. We will write that $\deg(S, R) = \deg_{G_S}(R)$. When $S = \emptyset$, we have that $G_S = \mathcal{H}$, so we will write, for convenience, that

$$\deg(S, R) = \deg_{G_S}(R) = \deg(S \cup R) \tag{2.5}$$

Furthermore, the number of edges in G_S is the equal to the degree of S, so we have that

$$|\mathcal{E}_S| = \deg(S) \tag{2.6}$$

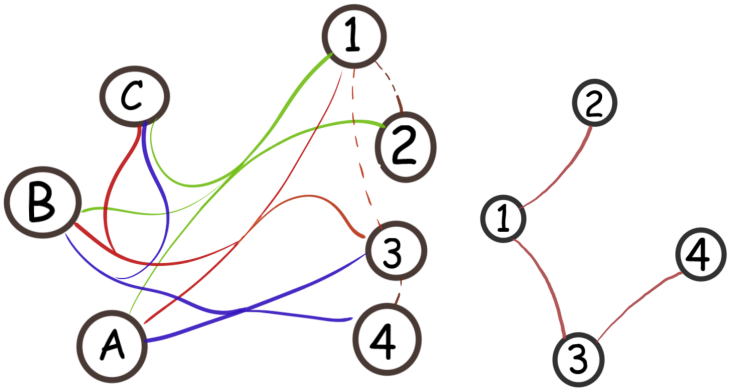

Figure 2.2: *Left*: $k = 5$ and $S = \{A, B, C\}$ *Right*: G_S

We can now impose an ordering on the vertices in $V \setminus S$ based on their S-codegree. If $u, v \in V \setminus S$, then we write that $u \prec_S v$ iff

$$\boxed{\deg_{G_S}(u) < \deg_{G_S}(v) \text{ or } \left(\deg_{G_S}(u) = \deg_{G_S}(v) \text{ and } \mathrm{id}(u) < \mathrm{id}(v)\right)} \tag{2.7}$$

This will become a very relevant definition in Chapter 5.

It is important to remark that the handshake lemma holds for codegrees in a uniform hypergraphs:

Lemma 2.1.1 (Generalized Handshake Lemma (GHL))**.** *Let $r \in [k-1]$. Then*

$$\sum_{S \in \binom{V}{r}} deg(S) = \binom{k}{r} m = O(m) \tag{2.8}$$

Proof. Each edge is counted $\binom{k}{r}$ times. □

Using this lemma for $r = 1$, we can generalize a simple lemma by Eden [ELRS15] about the number of vertices which come after a vertex in the previously defined ordering:

Lemma 2.1.2. *Let $S_u := \{w \in N_u \mid u \prec_\emptyset w\}$ be the set of neighbors of u that are after it in the total ordering of vertices based on degree ($S = \emptyset$). It is then true that*

$$\boxed{|S_u| = O\left(m^{\frac{1}{2}}\right)}$$

.

Proof. Using Lemma 2.1.1, we have for each $w \in S_u$ that $d_w \geq d_u \geq \frac{|N_u|}{k-1} \geq \frac{|S_u|}{k-1}$, so

$$km \overset{\text{GHL}}{\geq} \sum_{w \in S_u} d_w \geq |S_u| \left(\frac{|S_u|}{k-1}\right) \implies |S_u| = O(\sqrt{m})$$

□

Hypergraphs generalize graphs, but are also special cases of them. A 2-uniform hypergraph is a graph, and every hypergraph can be represented by a bipartite graph:

Definition 2.1.1. *If $\mathcal{H}$ is a hypergraph, then we define the **bipartite representation** of $\mathcal{H}$ as the bipartite graph $G_\mathcal{H} = (V; \mathcal{E}, R)$, where the edge $r = \{v, E\} \in R$ exists between a vertex v and a hyperedge E of $\mathcal{H}$ if $v \in E$.*

$G_\mathcal{H}$ will be a very important tool in how we study and understand hypergraphs.

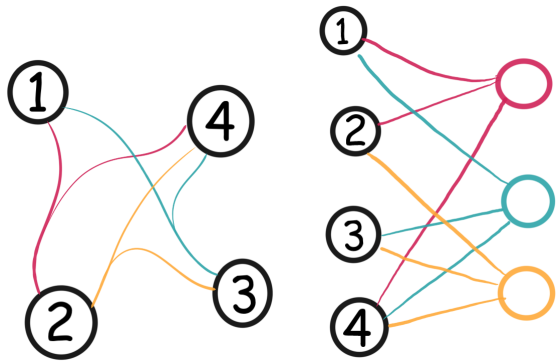

Figure 2.3: *Left*: A 3-uniform hypergraph, *Right*: Its bipartite representation.

2.2 Data Stream Algorithms

The streaming model dictates that the input to a streaming algorithm is a sequence of data points σ. The data points are drawn from some universe with N elements, with N being typically very large. The length of the stream is denoted by $|\sigma|$ and can be arbitrarily large as well.

In the streaming model, efficiency is measured in terms of the worst-case space complexity of the algorithm in question, which in turn is measured in terms of N and maybe $|\sigma|$. It is important to note that **we do not regard the time it takes for the algorithm to process each element in the stream as part of its complexity**, even though this is a very reasonable thing to consider when building algorithms in practice.

Within the streaming model itself, there are a few sub-models that are often considered. Typically, one encounters **insertion-only** streams, where data points cannot be deleted or altered. Indeed, this is the type of stream that this thesis will mostly deal with. However, many algorithms deal with dynamic datasets, where data points could be deleted or altered at any time. This streaming model is called the **turnstile model** and it is particularly useful in large networks where edges and vertices are created or deleted.

Streaming algorithms for graphs and hypergraphs assume a few more things about the input stream. The data points in the stream are **hyperedges**. We assume that no duplicate hyperedges are given, which means that $|\sigma| < |\mathcal{V}|^k$ for a k-uniform hypergraph. The complexity of the algorithms can be expressed using various possible hypergraph characteristics, such as the number of vertices, edges, cycles, paths of length-2 etc... The algorithms that will be covered in this thesis assume that the hypergraphs streamed are all uniform.

We will be making extensive use of the following boosting theorem from the field of Randomized Algorithms:

Theorem 2.2.1 (Median-of-Means improvement)**.**

If $\widehat{est}$ is an unbiased estimator of some statistic, then one can obtain an (ε, δ)-multiplicative estimate of that statistic using K independent samples of $\widehat{est}$, where:

$$K = \frac{C\,\mathbf{Var}[\widehat{est}]}{\left(\mathbf{Exp}[\widehat{est}]\right)^2} \cdot \frac{1}{\varepsilon^2} \cdot \ln\left(\frac{2}{\delta}\right)$$

where C is some constant.

We will also be needing the very useful and powerful Chernoff bounds to restrain the error probabilities in our algorithms:

Theorem 2.2.2 (Chernoff bounds)**.** *Let $X_1, ...X_n$ be n independent Bernoulli random variables. Let $X = \sum_{i=1}^{n} X_i$ and let $\varepsilon \in (0, 1)$. Then*

$$\Pr\left[X \geq (1+\varepsilon)\,\mathbf{Exp}[X]\right] \leq e^{-\frac{\varepsilon^2\,\mathbf{Exp}[X]}{3}}$$

$$\Pr\left[X \leq (1-\varepsilon)\,\mathbf{Exp}[X]\right] \leq e^{-\frac{\varepsilon^2\,\mathbf{Exp}[X]}{2}}$$

2.2.1 ℓ_0 sampling

Some of the algorithms that follow make use of ℓ_0 sampling as a streaming primitive. This section covers the necessary definitions and theorems that will be used.

Suppose that a stream σ was drawing elements from universe $U = [N]$. Let f_0 be the ℓ_0-norm of σ, i.e. the number of distict elements of σ. The goal is to uniformly sample an element of σ with probability $\frac{1}{f_0}$, or with probability as close to uniform as possible. We can tolerate a small probability of failure, but we would like our sampling, when successful, to be perfect (zero relative error)[1]

Very efficient algorithms have been proposed to solve the ℓ_0 sampling problem in the turnstile streaming model. Here we make use of the following result by Jowhari et al [JST11]:

Theorem 2.2.3 (Theorem 2 in [JST11]). *There exists a zero relative error ℓ_0 sampler which uses $O\left(\log^2 n \log\left(\frac{1}{\delta}\right)\right)$ bits of space and outputs a coordinate $i \in [N]$ uniformly from the support of σ with probability of success at least $1 - \delta$.*

2.2.2 Reservoir Sampling

Reservoir sampling is a widely-used method for sampling in one pass and uniformly at random an element from a data stream. The method is very simple: "the i-th item is sampled with probability $\frac{1}{i}$. If it is not sampled, the previously sampled item is retained." Under this scheme, the probability that any single element is ultimately sampled is $\frac{1}{|\sigma|}$, where $|\sigma|$ is the length of the stream.

2.2.3 F_0 estimation

Estimating the number of distinct elements (or F_0 frequency moment) is a very important problem in the theory of data streams. It has been proven by Kane, Nelson and Woodruff [KNW10] that an $(1 \pm \varepsilon)$-approximation to F_0 can be found in $O(\varepsilon^{-2} + \log n)$ bits of space in the streaming setting. We shall briefly use this result in Chapter 5.

2.3 Communication Complexity

Communication Complexity studies how efficiently certain functions can be communicated between one or more parties. Theorems and definitions from this beautiful field of Computer Science will be used later on when we establish lower bounds for some streaming algorithms.

Succinctly, according to the basic model of communication proposed by Andrew Yao, two players, Alice and Bob, possess two n-bit strings, x and y respectively. They wish to compute a function $f(x, y)$ by exchanging information. They both have access to unlimited computing resources, and their goal is to minimize the total amount of communication, in terms of bits, that they exchange.

Remark: *The classic communication model involves two parties, but many generalizations exist to involve more parties. The multi-party model that this thesis will use is called the **number-in-hand** (NOH) model. Alice, Bob and Charlie each hold an input string that is private to them. They communicate by writing messages on a **blackboard** that*

[1] There are efficient samplers that approximate uniformity very closely

everyone can see. The goal is for some party to produce a function of the three inputs. Another very popular multi-party communication model is the ***Number-In-Forehead*** *(NIF) model, where every party can see all the inputs except their own.*

A scheme for communicating a function f is called a **communication protocol**. A protocol can be deterministic, or randomized, and its randomness can be either public or private. The **cost** of a protocol is the number of bits exchanged between Alice and Bob in the worst case, taken over all inputs and all instances of the randomness that is possibly there[2]. Randomized communication protocols may also err with some probability. We say that a protocol has error ε if its error probability over its randomness for any input pair (x, y) is at most ε. We refer the interested reader to two textbooks ([KN96], [RY20]) on Communication Complexity for an in-depth exposition on the matter. We can now fully state the following definition.

Definition 2.3.1. *The* ***communication complexity*** $R_\varepsilon(f)$ *of a function* $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *is the worst-case cost of any protocol* P *that evaluates* f *with error probability at most* $\varepsilon \leq \frac{1}{3}$.

Remark: *The definitions of cost, error and complexity can be easily generalized for the multi-party model mentioned earlier.*

The Disjointness Communication Problem The disjointness function is arguably the most important one in the theory of communication complexity.

Definition 2.3.2. *Let* $X, Y \subseteq [n]$ *be two subsets of* $[n]$*, and let* $x, y \in \{0,1\}^n$ *be their characteristic bitstrings. The* ***disjointness*** *function* $DISJ_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *tells whether* X *and* Y *are disjoint:*

$$DISJ_n(x, y) = \begin{cases} 1, & \text{if } X \cap Y = \emptyset \\ 0, & \text{otherwise} \end{cases}$$

Remark: *The* $DISJ_n$ *function can be easily generalized for* t *parties:*

$$_tDISJ\ (x_1, ..., x_t) = 1 \text{ iff } \bigcap_{i=1}^{t} X_i = \emptyset$$

Recall that we are exclusively operating in the number-in-hand, blackboard model.

Communicating disjointness is hard, as reflected in the following theorem. This hardness result is a cornerstone of Communication Complexity theory and implies many hardness results in the field of streaming algorithms.

[2]Other measures of cost, such as the expected amount of communication also exist, but will not be of interest in this thesis

Theorem 2.3.1 (Hardness of Set-Disjointness [HW07])**.** *The communication complexity of set-disjointness is large:*

$$\boxed{R_\varepsilon(DISJ_n) = \Omega(n)}$$

There are a few *variants* to the classic set-disjoinness problem we saw above which will be of much use later on. All of the variants are hard communication problems, as can be easily shown through a reduction to Theorem 2.3.1:

1. **The ${}_t\mathbf{UDISJ}_n$ problem [CKS03]**: we are *promised* that the sets are *either* non-intersecting *or* their intersection has size 1.
2. **The $\mathbf{DISJ}_n^{r,T}$ problem [BOV13]**: we are *promised* that the sets have size r, and that if they have an intersection, its size will be at least T.

Theorem 2.3.2. *For the communication complexity of the above variants of the disjointness problem, it is known that:*

1. $R_\varepsilon({}_tUDISJ_n) = \Omega\left(\frac{n}{t}\right)$ *[CKS03]*
2. *For $r < n/2$, we have that $R_\varepsilon(DISJ_n^{r,T}) = \Omega\left(\frac{r}{T}\right)$ [BOV13]*

The Gap-Hamming Communication Problem

Definition 2.3.3. *Alice and Bob are given two n-bit strings $x, y \in \{0,1\}^n$, under the promise that the Hamming distance[3] $\Delta(x,y)$ between x and y satisfies either*

- $\Delta(x,y) \leq \frac{n}{2} - \sqrt{n}$, *or*
- $\Delta(x,y) \geq \frac{n}{2} + \sqrt{n}$

Their goal is to figure out which is the case.

This problem is another example of a hardness result in communication complexity, as shown below [RY20]

Theorem 2.3.3. *Any randomized protocol that solves the Gap-Hamming problem must have communication cost of $\Omega(n)$.*

It will be useful to generalize this problem to the multi-party model, something that, to our knowledge, has not yet been studied in the literature:

Definition 2.3.4. *When multiple parties are involved, the Hamming Distance of strings $x_1, ..., x_t$ can be defined as the number of positions in which all t strings differ.*

[3]The Hamming distance is the number of places in which x and y differ: $\Delta(x,y) = |\{i \in [n] \mid x_i \neq y_i\}|$

Theorem 2.3.4. *The communication cost of any randomized protocol solving the multi-party version of the Gap-Hamming problem is* $\Omega(n)$.

Proof. If a protocol existed to solve the problem in $o(n)$ communication, then that protocol could be used to solve the Gap-Hamming problem for 2 players in $o(n)$ communication. Alice would simply "play the part" of the first $t-1$ players, with all their inputs being equal to her input. □

2.4 Notation Appendix

This is a table of the most important symbols and notation that this thesis uses:

Symbol	Meaning
$\mathcal{H} = (V, \mathcal{E})$	A hypergraph
n	Number of vertices
m	Number of edges
$\mathcal{E}(S)$	Hyperedges within set $S \subseteq V$
$\mathcal{V}(S)$	Vertices inside a set S of hyperedges
$\deg(S)$	Degree of a vertex subset
N_S	Neighborhood of a vertex subset
$\deg(S, R)$	S-codegree of R
G_S	S-hypergraph of $\mathcal{H}$
$\deg_{G_S}(R)$	Degree of R in G_S
$u \prec_S v$	S-codegree ordering of vertices
$G_{\mathcal{H}}$	bipartite representation of $\mathcal{H}$
$R_\varepsilon(f)$	the randomized CC of f with error ε
S	The number of simplices of $\mathcal{H}$.

Chapter 3

Mathematical Insights

In this chapter, we extend tools from hypergraph theory and extremal graph theory, eventually developing certain novel mathematical results that will assist in the analysis of the algorithms to follow.

First, we use the concept of "hyper-forest decomposition" as explored by Frank and Kiraly [FKK03] to develop a natural generalization to the *arboricity* notion in graphs. We call this generalized arboricity notion the **hyper-arboricity** of a uniform hypergraph. This in turn leads to a sharp upper bound on a degree-related quantity in uniform hypergraphs. The methodology and definitions could be of independent interest and usefulness in other hypergraph applications.

Second, we generalize the folklore $S = O(m^{\frac{3}{2}})$ upper bound on the number of triangles in a graph to a $O(m^{\frac{k+1}{k}})$ upper bound for the number of simplices in a k-uniform hypergraphs. The basis of our technique is an inductive argument along with degree-based vertex partitioning, and is a method that we will use often for the remainder of this thesis.

3.1 Hyper-forest packing

In graphs a **forest** is an acyclic graph. Equivalently, forests are exactly the graphs in which for every subset X of the edge set, the number of incident vertices is strictly greater than $|X|$. In general hypergraphs we refer to this property as the **Strong Hall Property**.

Lemma 3.1.1 (**The Strong Hall property**). *A hypergraph $\mathcal{H} = (V, \mathcal{E})$ satisfies the Strong Hall property if and only if for any non-empty subset $X \subseteq \mathcal{E}$, we have that $|\mathcal{V}(X)| > |X|$. Equivalently, for any $S \subseteq V$, $|\mathcal{E}(S)| < |S|$.*

Acyclicity and the Strong Hall property coincide for regular graphs, but not for hypergraphs in general. In this work, we will use the Strong Hall property to characterize a family of hypergraphs which we will call *hyperforests* ([FKK03], [Lov68], [Lov70]):

Definition 3.1.1. *A hypergraph $\mathcal{H}$ is a **hyperforest** if and only if it satisfies the Strong Hall property.*

Seeking a generalization of the classic forest-packing theorems of Tutte and Nash-Williams ([**?**]), Frank and Kiraly [FKK03] use matroids to give a necessary and sufficient condition for the decomposition of a hypergraph into edge-independent sub-hyperforests.

Theorem 3.1.1. *The edge-set $\mathcal{E}$ of a hypergraph $\mathcal{H} = (V, \mathcal{E})$ can be decomposed into k hyperforests if and only if we have*

$$\mathcal{E}(X) \leq k(|X| - 1) \tag{3.1}$$

for all non-empty subsets $X \subseteq V$

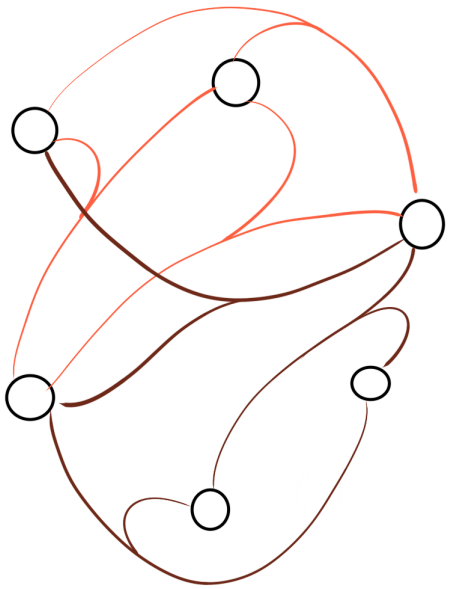

Figure 3.1: This 3-uniform hypergraph can be decomposed into 2 hyperforests. Its hyperarboricity is 1, so the hypergraph itself is hyperforest

Corollary 3.1.1. *Let $\rho(\mathcal{H})$ be the minimum number of edge-disjoint hyperforests that $\mathcal{H}$ can be decomposed into. We will call $\rho(\mathcal{H})$ the* ***hyperarboricity*** *of $\mathcal{H}$. Then:*

$$\rho(\mathcal{H}) = \max_{\mathcal{H}'=(V',\mathcal{E}')<\mathcal{H}} \left\lceil \frac{|\mathcal{E}'|}{|V'| - 1} \right\rceil \tag{3.2}$$

where $<$ denotes the sub-hypergraph relation.

Proof. Let $V' \subseteq V$ and consider an arbitrary sub-hypergraph $\mathcal{H}' = (V', \mathcal{E}')$ of the induced (by V') sub-hypergraph $\mathcal{H}'' = (V', \mathcal{E}'') < \mathcal{H}$. Then, $|\mathcal{E}'| \leq |\mathcal{E}''|$, so by (3.1)

$$\rho(\mathcal{H}) \geq \frac{|\mathcal{E}''|}{|V''| - 1} \geq \frac{|\mathcal{E}'|}{|V'| - 1}$$

Since $\rho(\mathcal{H})$ is an integer we arrive at our conclusion. □

3.2 Hyper-arboricity upper bound on 3-graphs

We follow the approach of Chiba and Nishizeki [CN85] to arrive at two upper bounds for the hyperarboricity of 3-uniform hypergraphs.

Theorem 3.2.1. *If $\mathcal{H} = (V, \mathcal{E})$ is a 3-uniform hypergraph with $n = |V|, m = |\mathcal{E}|$ and $n = O(m)$, then we have the following upper bound for the hyperarboricity of $\mathcal{H}$*

$$\rho(\mathcal{H}) = O(m^{2/3}) \tag{3.3}$$

Proof. Let $\mathcal{H}' < \mathcal{H}$ be such that $\rho(\mathcal{H}) = \left\lceil \frac{|\mathcal{E}(\mathcal{H}')|}{|V(\mathcal{H}')|-1} \right\rceil$. Such a sub-hypergraph $\mathcal{H}'$ must exist by Corollary 3.1.1. Let $p = |V(\mathcal{H}')|$, $q = |\mathcal{E}(\mathcal{H}')|$ and let $k = \binom{p}{3}$ be the number of edges on a complete 3-uniform hypergraph with p vertices. Observe that

$$6k = (p-1)^3 - (p-1) \tag{3.4}$$

We can distinguish between two cases:

- $k \leq m$. We have:

$$\rho(H) = \left\lceil \frac{q}{p-1} \right\rceil \leq \left\lceil \frac{k}{p-1} \right\rceil \leq \left\lceil \frac{p(p-2)}{6} \right\rceil \leq \left\lceil \frac{p^2}{6} \right\rceil$$

  By 3.4, we have that $p - 1 = [(6k + p - 1)]^{1/3}$, giving

$$\rho(H) \leq O(6k + p - 1)^{2/3} = O(6m + n - 1)^{2/3} = O(m^{2/3}) \tag{3.5}$$

  because $k \leq m, p \leq n$ and $n = O(m)$.

- $k \geq m$. By 3.4, we have that $6k \leq (p-1)^3$, so:

$$\rho(H) = \left\lceil \frac{q}{p-1} \right\rceil \leq \left\lceil \frac{m}{p-1} \right\rceil \tag{3.6}$$

$$\overset{m \leq k}{\leq} \left\lceil \frac{mk^{1/2}}{(p-1)^{3/2}} \right\rceil^{2/3} \tag{3.7}$$

$$= O\left[\left(\frac{m(p-1)^{3/2}}{(p-1)^{3/2}} \right)^{2/3} \right] = O(m^{3/2}) \tag{3.8}$$

□

Using Theorem 3.2.1 we can arrive at the following upper bound:

Theorem 3.2.2. *Let $\mathcal{H} = (V, \mathcal{E})$ be a 3-uniform hypergraph with $|\mathcal{E}| = m$ and $|V| =$*

$n = O(m)$. Then:

$$\sum_{\{u,v,w\}\in\mathcal{E}} \min\{d_u, d_v, d_w\} \leq 3m\rho(H) = O(m^{5/3}) \tag{3.9}$$

Proof. Consider a decomposition of $\mathcal{H}$ into $\rho(\mathcal{H})$ hyperforests $\mathcal{F}_i = (V_i, \mathcal{E}_i)$ for $i \in [\rho(\mathcal{H})]$. In each hyperforest $\mathcal{F}_i$, we will associate each hyperedge $E \in \mathcal{E}_i$ with a *representative vertex* $v^{(i)}(E) \in E$ such that no two hyperedges in $\mathcal{E}_i$ share a representative. We can find such a mapping from $\mathcal{E}_i$ to V_i if and only if $G_{\mathcal{F}_i}$ has a perfect matching that saturates $\mathcal{E}_i$. Since $\mathcal{F}_i$ is a hyperforest, Lemma 3.1.1 and Hall's Matching Theorem ensure that a perfect matching saturating $\mathcal{E}_i$ exists. Using equation 3.10 and the handshake lemma, we get:

$$\begin{aligned}
\sum_{\{u,v,w\}\in\mathcal{E}} \min\{d_u, d_v, d_w\} &= \sum_{1\leq i\leq\rho(\mathcal{H})} \sum_{\{u,v,w\}\in\mathcal{E}_i} \min\{d_u, d_v, d_w\} \\
&\leq \sum_{1\leq i\leq\rho(\mathcal{H})} \sum_{E\in\mathcal{E}_i} d_{v^{(i)}(E)} \\
&\leq \sum_{1\leq i\leq\rho(\mathcal{H})} \sum_{v\in V} d_v = 3m\rho(\mathcal{H}) = O(m^{5/3})
\end{aligned}$$

□

Remark: *This upper bound is tight, and realized by the case of the 3-uniform hypergraph*

3.2.1 Generalizing to k-graphs

The previous findings can be generalized to k-uniform hypergraphs:

Theorem 3.2.3. *If $\mathcal{H} = (V, \mathcal{E})$ is a k-uniform hypergraph with $n = |V|, m = |\mathcal{E}|$ and $n = O(m)$, then we have the following upper bound for the hyperarboricity of $\mathcal{H}$*

$$\rho(\mathcal{H}) = O(m^{\frac{k-1}{k}}) \tag{3.10}$$

Proof. The proof follows along similar lines to Theorem 3.2.1. We let $\mathcal{H}' < \mathcal{H}$ be such that $\rho(\mathcal{H}) = \left\lceil \frac{|\mathcal{E}(\mathcal{H}')|}{|V(\mathcal{H}')|-1} \right\rceil$. Let $p = |V(\mathcal{H}')|$, $q = |\mathcal{E}(\mathcal{H}')|$ and let $s = \binom{p}{k}$ be the number of edges on a complete k-uniform hypergraph with p vertices. We make two algebraic observations:

1. $\mathbf{p = O\left((s + p)^{1/k}\right)}$. We know that $s \geq Cp^k \geq Cp^k - p$ for some constant C, which implies that $s + p = \Omega(p^k)$.
2. $\mathbf{k! \cdot s = O((p - 1)^k)}$. We have: $k! \cdot s = p(p-1)\cdots(p-k+1) \leq p(p-1)^{k-1} = (p-1)^k + (p-1)^{k-1} = O((p-1)^k)$

Now, as before, we can distinguish between two cases:

- $s \leq m$. We have:

$$\rho(H) = \left\lceil \frac{q}{p-1} \right\rceil \leq \left\lceil \frac{s}{p-1} \right\rceil \leq \left\lceil \frac{p(p-2)\cdots(p-k+1)}{k!} \right\rceil \leq \left\lceil \frac{p^{k-1}}{k!} \right\rceil = O(p^{k-1})$$

  But we have that $p = O\left((s+p)^{1/k}\right)$, giving

$$\rho(H) = O\left((s+p)^{\frac{k-1}{k}}\right) = O\left((m+n)^{\frac{k-1}{k}}\right) = O(m^{\frac{k-1}{k}})$$

  because $s \leq m, p \leq n$ and $n = O(m)$.

- $s \geq m$. By the second observation above, we get:

$$\begin{aligned}\rho(H) = \left\lceil \frac{q}{p-1} \right\rceil &\leq \left\lceil \frac{m}{p-1} \right\rceil \\ &\overset{m\leq s}{\leq} \left\lceil \frac{ms^{\frac{1}{k-1}}}{(p-1)^{\frac{k}{k-1}}} \right\rceil^{\frac{k-1}{k}} \\ &= O\left[\left(\frac{m(p-1)^{\frac{k}{k-1}}}{(p-1)^{\frac{k}{k-1}}} \right)^{\frac{k-1}{k}} \right] = O(m^{\frac{k-1}{k}})\end{aligned}$$

□

As an immediate corollary to this generalized bound, we can get the following generalization of Theorem 3.2.2:

Theorem 3.2.4. *Let $\mathcal{H} = (V, \mathcal{E})$ be a k-uniform hypergraph with $|\mathcal{E}| = m$ and $|V| = n = O(m)$. Then:*

$$\sum_{\{u_1,\ldots,u_k\}\in\mathcal{E}} \min\{deg(u_1), \ldots, deg(u_k)\} \leq km\rho(H) = O\left(m^{2-\frac{1}{k}}\right) \tag{3.11}$$

Proof. We omit the proof because it is almost identical to the proof of theorem 3.2.2. □

Remark: *As before, this bound is tight. Equality holds for complete k-graphs.*

3.3 Bounding the simplex-count in uniform hypergraphs

In a graph G with m edges and T triangles, it is always true that $T = O(m^{3/2})$. We can generalize this inequality to k-uniform hypergraphs, starting with $k = 3$.

Theorem 3.3.1. *Let $\mathcal{H} = (V, \mathcal{E})$ be a 3-graph with S simplices and m edges. Then*

$$S = O\left(m^{\frac{4}{3}}\right)$$

Proof. Let $v \in V$. We distinguish between two cases:

1. *v is heavy:* $\mathbf{deg(v) > m^{2/3}}$. By the handshake lemma we have that $\sum_v \deg_v = 3m$, so there are at most $(3m)^{1/3}$ heavy vertices. Each vertex can participate in at most m simplices, so the number of simplices with at least one heavy vertex is at most $(3m)^{1/3} \cdot m = O(m^{4/3})$.

2. *v is light:* $\mathbf{deg(v) \leq m^{2/3}}$. We take out from $\mathcal{H}$ all the heavy vertices, forming the hypergraph $\mathcal{H}' = (V', \mathcal{E}')$. This can only decrease the degree of any light vertex. Let S' be the total number of simplices in $\mathcal{H}'$. Then, the number of simplices in $\mathcal{H}$ consisting of only light vertices is equal to S'.

   If $\{v, x, y, z\}$ is a simplex in $\mathcal{H}'$, then the collection of edges $\{\{v, x, y\}, \{v, y, z\}, \{v, x, z\}\}$ is called a **3-wedge centered at v**. Let W_v be the number of 3-wedges centered at vertex v. We have that

   $$S' \leq \frac{1}{4} \sum_{v \in V'} W_v$$

   because each simplex is counted 4 times. The key observation to make is that:

   Observation 3.3.2. *In the graph G_v, a 3-wedge centered at v is a triangle, so W_v is equal to the number of triangles in G_v*

   This implies that $W_v = O(\deg(v)^{3/2})$, because $|\mathcal{E}(G_v)| = \deg(u)$. Since $\deg(v) \leq m^{2/3}$ for each $v \in V'$, we have:

   $$\begin{aligned} S' = O\left(\sum_{v \in V'} \deg(v)^{3/2}\right) &= O\left(\sum_{v \in V'} \deg(v)^{1/2} \cdot \deg(v)\right) \\ &= O\left(m^{1/3} \sum_{v \in V'} \deg(v)\right) = O(m^{4/3}) \end{aligned}$$

□

Remark: *This upper bound is tight: On a complete 3-uniform hypergraph with n vertices, there are $\Theta(n^4)$ simplices and $\Theta(n^3)$ edges.*

From here, we can use an inductive argument to show a generalized upper bound on the number of simplices for k-uniform hypergraphs:

Theorem 3.3.3. *Let $\mathcal{H} = (V, \mathcal{E})$ be a k-uniform hypergraph with S simplices and $|\mathcal{E}| = m$. Then*

$$S = O\left(m^{\frac{k+1}{k}}\right)$$

Proof. We induct on k. For $k = 3$ we have already proven our claim with Theorem 3.3.3. Now we let $k > 3$. For $v \in V$, we again distinguish between two cases:

- $\mathbf{deg(v) > m^{\frac{k-1}{k}}}$ (heavy vertices): As before, due to the handshake lemma, there are $O(m^{1/k})$ heavy vertices, each of which can be in at most m simplices, which gives the desired bound.
- $\mathbf{deg(v) \leq m^{\frac{k-1}{k}}}$ (light vertices): Let $\mathcal{H}' = (\mathcal{V}', \mathcal{E}')$ be $\mathcal{H}$ with all light vertices removed. Let $\mathcal{H}'$ contain S' simplices (composed only of heavy vertices in $\mathcal{H}$). Finally, let S'_v is the number of simplices in $\mathcal{H}'$ in which a heavy vertex v is involved. Each such simplex corresponds to a simplex in G_v. By the inductive hypothesis, $|S'_v| = O\left(\deg(v)^{\frac{k}{k-1}}\right)$ because $|\mathcal{E}_v| = \deg(v)$, so we get that:

$$\begin{aligned} S' = O\left(\sum_{v \in \mathcal{V}'} \deg(v)^{\frac{k}{k-1}}\right) &= O\left(\sum_{v \in \mathcal{V}'} \deg(v)^{\frac{1}{k-1}} \cdot \deg(v)\right) \\ &= O\left(m^{\frac{1}{k}} \sum_{v \in \mathcal{V}'} \deg(v)\right) = O(m^{\frac{k+1}{k}}) \end{aligned}$$

□

Remark: *As before, this is a tight bound. On a complete k-uniform hypergraph with n vertices there are $\Theta(n^{k+1})$ simplices and $\Theta(n^k)$ edges.*

Chapter 4

Sampling and Counting Triangles

This chapter is devoted to counting and sampling triangles from streams of undirected graphs. Although this problem has been widely studied (see the Introduction), the algorithms we present in this section are almost-optimal and not yet, to our knowledge, fully described in the literature.

First, we will be using the idea of *degree-based vertex partitioning* outlined in [KMPT12] to develop an "importance-sampling"-like technique for counting triangles. This technique was first conceived and used for triangle counting in [KMPT12], but the authors of that paper do not use it to develop an explicit algorithm that works in the general, arbitrary order, graph stream model. We complete that body of work by giving a $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$ algorithm for counting triangles in the streaming model.

Second, we use, almost without no additional work, the previously mentioned triangle-counting algorithm to *uniformly sample* triangles from a graph stream using $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$ bits of space in the worst case. This algorithm beats, to our knowledge, the best so far triangle sampling algorithm given by [PTTW13] which runs in $\widetilde{O}\left(\frac{mn}{T}\right)$-space in the worst case.

4.1 Counting Triangles in the Streaming Model

Our algorithm essentially tries to sample triples $\{u, v, w\}$ from a "small" set U which contains all possible triangles. If any triple were allowed to be sampled, then we would need to perform $O(n^3)$ samples in the worst case, so we use an "importance-sampling" method to narrow down our search space. Instead of traditional triples $\{u, v, w\}$, the triples in U are labeled by a vertex, so the elements of U are of the form $(u, \{v, w\}) \in V \times \binom{V}{2}$.

We construct U implicitly, using the degree-based partitioning idea introduced in [KMPT12]. U is made up of two disjoint sets of labeled triples: triples where the label is a high-degree (heavy) vertex and triples where it is low-degree. Formally:

$$U_L = \{(u, \{v, w\}) \mid \deg(u) < \sqrt{m} \text{ and } \{v, w\} \subseteq N_u\} \tag{4.1}$$

$$U_R = \{(u, \{v, w\}) \mid \deg(u) \geq \sqrt{m} \text{ and } \{v, w\} \in E\} \tag{4.2}$$

$$U = U_L \cup U_R \tag{4.3}$$

We let $L = \{v \in V \mid \deg(\text{v}) < \sqrt{m}\}$ be the set of **light** vertices and $H = V \setminus L$ be the

heavy vertices of V. The number of candidate triples in U_L is $|U_L| = \sum_{v \in L} \binom{\deg(v)}{2}$ and the number of triples containing at least one heavy vertex is $|U_H| = |H| \cdot m$

It is critical in the analysis of our algorithm that labeled triples are sampled from U uniformly. To achieve that, we use an "importance sampling" method. In one pass of preprocessing, we calculate the degree of every vertex. This also allows us to precisely calculate $|U_L|$ and $|U_H|$. For elements in U_L, we use the degree information to sample low-degree vertices $v \in L$ with a probability proportional to the number of triples in U_L labeled with v. For elements in U_H, we just need to sample a vertex $v \in H$ and an edge $e \in E$ uniformly at random.

The proposed triangle counting algorithm is shown below:

Algorithm 1: Counting triangles in the streaming model

procedure DEGREE-BASED-TRIANGLE-COUNT(σ)
  Pass 1: Calculate the degree $\deg(v)$ for every vertex $v \in V$.
  Offline: Calculate U_L and U_H based on the degree information.

  Sample a pair $(v, \{w, z\}) \in V \times V^2$**,** s **times in parallel, as follows:**
    $F \leftarrow$ Flip a coin with probability $\frac{|U_L|}{|U_L|+|U_H|}$.
    if $F =$ HEADS **then**
      Sample $v \in L$ with probability $\frac{\binom{\deg(v)}{2}}{|U_L|}$.
      Pass 2: Sample two different vertices w, z uniformly from $N(v)$.
    else
      Sample $v \in H$ with probability $\frac{1}{|H|}$.
      Pass 2: Sample some edge (w, z) with probability $\frac{1}{m}$
    end if
    Pass 3:
      Check if $\{v, w, z\}$ forms a triangle.
      Let $X_i = 1$ if that is the case, with $1 \leq i \leq s$. Otherwise set $X_i = 0$.
  Output: $\frac{1}{s} \sum_{i=1}^{s} X_i$.
end procedure

4.1.1 Analysis

The following theorem is the gist of the algorithm developed above:

Theorem 4.1.1. *There exists a 3-pass streaming algorithm for $(\varepsilon, n^{-O(1)})$-estimating the triangle count in an undirected graph, using $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$ bits of space in the worst case.*

We prove that our algorithm is correct, accurate and memory-efficient through a series of concise lemmata:

Lemma 4.1.1. $|U| = |U_L \cup U_R| = O(m^{3/2})$

Proof. $|U| = |U_L \cup U_H| = |U_L| + |U_H|$. We have that

$$|U_L| = \sum_{u \in L} \binom{\deg(u)}{2} < \sum_{u \in L} \deg(u)^2 = O(m^{3/2}) \tag{4.4}$$

as $\deg(u) < \sqrt{m}$ for $u \in L$. Also, $|U_H| \leq \sqrt{2m} \cdot m = O(m^{3/2})$ as $|H| \leq \sqrt{2m}$ by the handshaking lemma. □

Lemma 4.1.2. *Every triangle is found 3 times in U as an unlabeled triple $\{u, v, w\}$*

Proof. If a triangle $\{u, v, w\}$ contains i light vertices and $3-i$ heavy vertices, with $0 \leq i \leq 3$, then it appears i times in U_L and $3 - i$ times in U_H, for a total of 3 times. □

Lemma 4.1.3. *Our algorithm samples a labeled triple **uniformly** from U.*

Proof. If $(u, \{v, w\}) \in U_L$, then it is sampled with probability

$$\frac{\cancel{|U_L|}}{|U_L| + |U_H|} \cdot \frac{\cancel{\binom{\deg(u)}{2}}}{\cancel{|U_L|}} \cdot \frac{1}{\cancel{\binom{\deg(u)}{2}}} = \frac{1}{|U_L| + |U_H|} = \frac{1}{|U|} \tag{4.5}$$

If $(u, \{v, w\}) \in U_H$, then it is sampled with probability

$$\frac{\cancel{|U_H|}}{|U_L| + |U_H|} \cdot \frac{1}{\cancel{|H|}} \cdot \frac{1}{\cancel{m}} = \frac{1}{|U_L| + |U_H|} = \frac{1}{|U|} \tag{4.6}$$

□

Lemma 4.1.4. *If $s = \Theta\left(\frac{|U| \log n}{\varepsilon^2 T}\right)$, then Algorithm 1 (ε, n^{-d})-approximates T_3, where d is some constant greater than 1.*

Proof. First, it is easy to see that $\mathbb{E}[X_i] = \frac{3T}{|U|}$. Now, applying a standard Chernoff bound and a trivial union bound, we get that:

$$\Pr\left[\left|\frac{1}{s}\sum_{i=1}^{s} X_i - \frac{3T}{|U|}\right| \geq \varepsilon \cdot \frac{3T}{|U|}\right] \leq 2\exp\left(-\frac{\varepsilon^2 T s}{2|U|}\right) \leq n^{-d} \tag{4.7}$$

□

Lemma 4.1.5. *Algorithm 1 uses $\widetilde{O}\left(n + \frac{m^{3/2}}{T}\right)$ bits of space in the worst case.*

Proof. The algorithm uses $\widetilde{O}(n + s)$ bits of space. However, by Lemma 5.3.2, we have that $s = \widetilde{\Theta}\left(\frac{m^{3/2}}{T}\right)$ gives a good approximation. □

4.2 Triangle Sampling

We can make use of Algorithm 1 to make a method that samples a triangle from a graph stream uniformly using little memory.

When running Algorithm 1 but only sampling a single labeled triple ($s = 1$), the probability of sampling a triangle is $p = \frac{3T}{|U|}$. Given that a triangle is sampled, it is sampled uniformly from the set of all triangles in the graph, because the labeled triples are sampled uniformly (Lemma 4.1.3).

We will show that Algorithm 2 finds a triangle with probability at least $1-\varepsilon$. Even though the instructions in Lines 6-18 happen in parallel, we assume that they finish in some arbitrary order - a logical assumption for any computer system. In this order, let M be the number of the iterations that it takes for a triangle to be first detected. M is geometrically distributed with parameter $p = \frac{3T}{|U|}$, so we *expect* the that after $\mathbf{Exp}[M] = \frac{1}{p} = \frac{|U|}{3T} = O\left(\frac{m^{3/2}}{T}\right)$ iterations, a triangle will be found. Therefore, running Lines 6-18 in parallel only $\mathbf{Exp}[M]$ times should be enough to uniformly sample a triangle.

Algorithm 2: Uniform Triangle Sampling in the Streaming model

1: **procedure** DEGREE-BASED-TRIANGLE-SAMPLE(σ, ε)
2: **Pass 1**: Calculate the degree $\deg(v)$ for every vertex $v \in V$.
3: **Offline:** Calculate $|U_L|$ and $|U_H|$ based on the degree information.
4:
5: Execute $\frac{1}{\varepsilon} \cdot \frac{|U|}{3T}$ times **in parallel** :
6: $F \leftarrow$ Flip a coin with probability $\frac{|U_L|}{|U_L|+|U_H|}$.
7: **if** $F =$ HEADS **then**
8: Sample $v \in L$ with probability $\frac{\binom{\deg(v)}{2}}{|U_L|}$.
9: **Pass 2:** Sample two different vertices w, z uniformly from $N(v)$.
10: **else**
11: Sample $v \in H$ with probability $\frac{1}{|H|}$.
12: **Pass 2:** Sample some edge (w, z) with probability $\frac{1}{m}$
13: **end if**
14: **Pass 3**:
15: Check if $\{v, w, z\}$ forms a triangle.
16: **if** a triangle $t = \{v, w, z\}$ is found **then**
17: Output t
18: ▷ If multiple triangles are found, then any of them is returned
19: **end if**
20: Output FAIL.
21: **end procedure**

However, we want to upper bound (by ε) the probability that more parallel samplings are needed. By the Markov inequality, the probability that M actually becomes larger than $\frac{1}{\varepsilon} \cdot \frac{|U|}{3T}$ is at most:

$$\Pr\left[M \geq \frac{1}{\varepsilon} \cdot \frac{|U|}{3T}\right] \leq \frac{\mathbf{Exp}[M]}{\frac{1}{\varepsilon}\mathbf{Exp}[M]} = \varepsilon \tag{4.8}$$

Therefore, $\frac{1}{\varepsilon} \cdot \frac{|U|}{3T}$ parallel samplings are enough to guarantee with probability at least $1 - \varepsilon$ that a triangle will be found. Thus, we have proven the following theorem:

Theorem 4.2.1. *There is a 3-pass, streaming algorithm for sampling a triangle from an undirected graph uniformly at random. The algorithm uses* $O\left(n + \frac{m^{3/2}}{\varepsilon T}\right)$ *bits of space.*

Chapter 5

Simplex Counting Algorithms

In this chapter, we will be exploring the main problem of this thesis: counting simplices in hypergraph streams. We will be presenting a few algorithms to solve this problem, eventually finding an optimal algorithm, as claimed in Theorem 1.3.1. To illustrate our algorithms, we will first present them for 3-uniform hypergraphs and then generalize them for k-uniform hypergraphs.

The first algorithm we see is a straightforward generalization of a method in [BC17] that runs in $\widetilde{O}(m^{5/3}/S)$ for 3-graphs, and in $\widetilde{O}(m^{2-\frac{1}{k}}/S)$ in general. That algorithm is of particular interest because its analysis is not straightforward; indeed it prompted the use of hyperforest decomposition methods, as covered in Chapter 3, which may be of independent interest.

However, the $\widetilde{O}(m^{2-\frac{1}{k}}/S)$ upper bound is not optimal. By adapting Theorem 4.1.1, we present an algorithm for $k = 3$ that runs in $\widetilde{O}(n^2 + m^{4/3}/S)$ or in $\widetilde{O}(m^{4/3}/S)$-space given a codegree oracle. This algorithm can be adapted to sampling simplices uniformly from a 3-uniform hypergraph, akin to theorem 4.2.1.

Finally, we return to the method in [BC17] and generalize it in a different way, eventually obtaining an optimal $\widetilde{O}(m^{1+\frac{1}{k}}/S)$ algorithm that does not require codegree oracles. We note that, for our purposes, k is a constant, so it does not affect our analysis.

For the purpose of completeness, at the end of the chapter, we present some other, less efficient generalizations of triangle counting algorithms into the problem of simplex counting.

5.1 A first attempt at simplex counting

In their 2017 paper [BC17], Chakrabarti and Bera propose a generic algorithm for counting substructures in an arbitrary-order graph stream. The algorithm samples a sufficient number of edges, hoping to select an edge cover for the requested substructure being counted. Assuming that the size of the requested substructure is a constant, they use two passes to produce an unbiased estimator with bounded variance. The authors improve on this bound by proposing 4-pass streaming algorithms for counting cliques and cycles of odd size.

In the following sections we generalize their streaming algorithms to count simplices in 3-uniform hypergraphs. To that end, we use the insights on hyperforest decomposition covered in Chapter 3.

5.1.1 The Algorithm

We impose a **total ordering** on V according to Lemma 2.1.2. We associate a simplex on vertices u, v, w, z with the pair $((u, v, w), z) \in \binom{V}{3} \times V$ where:

- $\{u, v, w\} \in \mathcal{E}$.
- $u \prec_\emptyset v \prec_\emptyset w \prec_\emptyset z$, where recall that $\prec_\emptyset$ is our degree-based ordering.

In other words, we produce a **label** for each simplex in the graph. Given the label, then we can determine exactly the simplex referenced and each simplex has a unique label. We let S denote the number of simplices in $\mathcal{H}$ and the number of the simplices whose label contains the hyperedge $\{u, v, w\}$ is denoted by $S_{\{u,v,w\}}$.

Algorithm 3: Counting simplices in 3-uniform hypergraph streams

1: **procedure** 4-SIMPLEX-STREAM-COUNT(σ)
2: **Pass 1** ▷ $O(1)$ space
3: Pick an edge $\{u, v, w\} \in \mathcal{E}$ using reservoir sampling.
4: **Pass 2** ▷ $O(1)$ space
5: Calculate the degrees $\deg(u), \deg(v)$ and $\deg(w)$.
6: **Pass 3** ▷ $O(S(F_0) + rS(L_0))$ space
7: Let $r := \lceil \min\{\deg(u), \deg(v), \deg(w)\} \cdot m^{-1/2} \rceil$.
8: Re-arrange (if needed) u, v, w so that $u \prec_\emptyset v \prec_\emptyset w$.
9: **for** $k = 1$ to r **do**:
10: $Z_k \leftarrow 0$
11: Construct a virtual stream σ_k consisting of the vertices in $N(u) \setminus \{v, w\}$.
12: Sample vertex x_k from $N(u) \setminus \{v, w\}$ u.a.r using ℓ_0-sampling on σ_k
13: **end for**
14: Simultaneously, approximate $|N(u)|$ using F_0-estimation (▷ $O(\log n)$ space).
15: **Pass 4** ▷ $O(r)$ space
16: Compute the degrees $\deg(x_1), \deg(x_2), ..., \deg(x_r)$.
17: **for** $k = 1$ to r **do**
18: **if** $w \prec_\emptyset x_k$ **and** $\{u, v, w, x_k\}$ form a 4-simplex **then**
19: $Z_k = |N(u)| - 2$ ▷ u has the smallest degree
20: **end if**
21: **end for**
22: $Y \leftarrow \frac{1}{r}\left(\sum_{k=1}^{r} Z_k\right)$ ▷ Average out the trials
23: **return** $X = mY$ ▷ Scale to get an unbiased estimator
24: **end procedure**

The algorithm proceeds by sampling a hyperedge $\{u, v, w\}$ u.a.r. It then samples a vertex in the neighborhood of u u.a.r. and examines if the assembled label makes up a simplex. Necessary degree bookkeeping requires 4 passes for all the information to be collected.

A caveat that immediately appears in this case and not in the original algorithm is the uniform sampling of the vertex in the neighborhood of u (Line 11). In graphs, reservoir

sampling suffices because sampling a neighbor of u is equivalent to sampling uniformly an edge that is incident on u. The same is not true for hypergraphs, because sampling a hyperedge that contains u u.a.r. gives a probabilistic advantage to vertices which share many hyperedges with u. Thus, we need to use ℓ_0-sampling techniques in a substream consisting of neighbors of u.

A second caveat that appears is in the calculation of the size of the neighborhood of u. Treating every edge containing u as a virtual stream σ_u, we can calculate $|N(u)|$ by counting the distinct elements in σ_u. This can be done either precisely, in $O(n)$ space, or approximately in $O(\log n)$ space, using F_0-estimation algorithms.

5.1.2 Analysis

Our analysis proceeds like [BC17]. The only differences are the presence of ℓ_0-sampling, F_0 estimation, and the use of the upper bounds established in 3.2.

Remark: *In the analysis below, we will assume that the algorithms for ℓ_0 sampling and F_0 estimation produce the desired outputs correctly. Any precision errors are assumed to be absorbed into the quality of the estimator X. Our analysis does not examine the implications of this assumption in a more pedantic way because this algorithm is meant as a first-attempt to a suboptimal solution.*

Unbiased Estimator First we prove that our estimator X is unbiased. Let $\mathcal{E}_E = \mathcal{E}_{\{u,v,w\}}$ denote the event in which the hyperedge $E = \{u, v, w\}$ is sampled in Pass 1. Because we use ℓ_0 sampling in Line 12 of our algorithm above, we choose with probability $(|N(u)|-2)^{-1}$ each of the $S_{\{u,v,w\}}$ vertices which complete a valid simplex label with hyperedge E. Therefore, we have that

$$\mathbb{E}[Z_k \mid \mathcal{E}_{\{u,v,w\}}] = S_{\{u,v,w\}} \frac{1}{|N(u)| - 2}(|N(u)| - 2) = S_{\{u,v,w\}} \tag{5.1}$$

and so by the law of total expectation we get that

$$\mathbb{E}[X] = \frac{m}{r} \sum_{k=1}^{r} \sum_{\{u,v,w\} \in \mathcal{E}} \frac{1}{m} \mathbb{E}[Z_k \mid \mathcal{E}_{\{u,v,w\}}] = \frac{1}{r} \sum_{k=1}^{r} \sum_{\{u,v,w\} \in \mathcal{E}} S_{\{u,v,w\}} = S \tag{5.2}$$

Bounded variance Now we attempt to bound the variance. Proceeding along similar lines as above, we have that

$$\mathbb{E}[Z_k^2 \mid \mathcal{E}_{\{u,v,w\}}] = (|N(u)| - 2) \cdot S_{\{u,v,w\}} \leq |N(u)| \cdot S_{\{u,v,w\}} \tag{5.3}$$

$$\mathbb{E}[Z_{k_1} Z_{k_2} \mid \mathcal{E}_{\{u,v,w\}}] = S_{\{u,v,w\}}^2 \tag{5.4}$$

because Z_{k_1} and Z_{k_2} are independent events even when conditioned on $\mathcal{E}_{\{u,v,w\}}$. So we have:

$$\mathbb{E}[Y^2 \mid \mathcal{E}_{\{u,v,w\}}] = \mathbb{E}\left[\left(\frac{1}{r}\sum_{k=1}^{r} Z_k\right)^2 \mid \mathcal{E}_{\{u,v,w\}}\right] \tag{5.5}$$

$$= \frac{1}{r^2}\sum_{k=1}^{r}\mathbb{E}[Z_k^2 \mid \mathcal{E}_{\{u,v,w\}}] + \frac{1}{r^2}\sum_{k_1\neq k_2}\mathbb{E}[Z_{k_1}Z_{k_2} \mid \mathcal{E}_{\{u,v,w\}}] \tag{5.6}$$

$$\leq \frac{|N(u)|}{r}S_{\{u,v,w\}} + \frac{\binom{r}{2}}{r^2}S^2_{\{u,v,w\}} \leq 2\sqrt{m}S_{\{u,v,w\}} + S^2_{\{u,v,w\}} \tag{5.7}$$

because $\frac{|N(u)|}{r} \leq \frac{2\deg(u)}{r} \leq 2\sqrt{m}$ by the definition of r. Finally we get that:

$$Var[X] = m^2 Var[Y] \leq m^2\mathbb{E}[Y^2] \tag{5.8}$$

$$= m^2 \sum_{\{u,v,w\}\in\mathcal{E}} \frac{1}{m}\mathbb{E}[Y^2 \mid \mathcal{E}_{\{u,v,w\}}] \tag{5.9}$$

$$\leq 2m^{3/2}\sum_{\{u,v,w\}\in\mathcal{E}} S_{\{u,v,w\}} + m\sum_{\{u,v,w\}\in\mathcal{E}} S^2_{\{u,v,w\}} \tag{5.10}$$

$$\leq 2m^{3/2}S + m\sum_{\{u,v,w\}\in\mathcal{E}} S^2_{\{u,v,w\}} \tag{5.11}$$

At this point we need a bound on $\sum T^2_{\{u,v,w\}}$. Using Lemma 2.1.2, we argue that

$$\sum_{\{u,v,w\}\in\mathcal{E}} S^2_{\{u,v,w\}} = O(S\sqrt{m}) \tag{5.12}$$

because for any edge $\{u,v,w\}$ we have that $S_{\{u,v,w\}} \leq |S_u| = O(\sqrt{m})$, where we use S_u as in the proof of Lemma 2.1.2. This allows us to claim that

$$\boxed{Var[X] = O(m^{3/2}S)} \tag{5.13}$$

Space Complexity Using the *median-of-means* improvement, we can see that our algorithm uses space $O(m^{3/2}B/S)$ to output a $(\varepsilon, 1/3)$-approximation of T, where $B = O(S(F_0) + rS(L_0))$ is the space used by one instance of Algorithm 3. We must now verify that our algorithm does not cause B to explode.

Remark: *Let $S(F_0)$ be the space complexity of the F_0-estimator and $S(L_0)$ be the space complexity of the ℓ_0 sampler. We assumed that those algorithms function perfectly. To boost this confidence probability over all repetitions of Algorithm 3, the confidence parameters in each algorithm need to be adjusted accordingly. However, since the confidence parameter is inside a logarithm, it becomes insignificant in the total space complexity. We omit more thorough investigation because it escapes the purposes of this section.*

We calculate the expected value of B: $\mathbb{E}[B] = S(F_0) + S(L_0)\mathbb{E}[r]$. We have that:

$$\mathbb{E}[r] = \frac{1}{m} \sum_{\{u,v,w\} \in \mathcal{E}} \left\lceil \frac{\min\{\deg(u), \deg(v), \deg(w)\}}{\sqrt{m}} \right\rceil \tag{5.14}$$

$$= O\left(m^{-3/2} \sum_{\{u,v,w\} \in \mathcal{E}} \min\{\deg(u), \deg(v), \deg(w)\} \right) \tag{5.15}$$

Using Theorem 3.2.2 we get that $\mathbb{E}[r] = O(m^{5/3-3/2}) = O(m^{1/6})$. Thus $B = O(S(F_0) + m^{1/6} S(L_0))$ in expectation. We can assume that $S(F_0) = \widetilde{O}(\log n)$ and $S(L_0) = \widetilde{O}(\log^2 n)$, meaning that

$$\mathbb{E}[B] = \widetilde{O}(\log_n + m^{1/6} \log^2 n) = \widetilde{O}(m^{1/6} \log^2 n) \tag{5.16}$$

If we terminate the algorithm when it uses more than (say) $10m^{1/6} \log^2 n$ space, then with high probability we get a worst-case memory bound of $O(m^{5/3} \log^2 n/S)$. We thus arrive to the following theorem:

Theorem 5.1.1. *If $\mathcal{H} = (V, \mathcal{E})$ is a 3-uniform hypergraph with n vertices and m edges, then a 4-pass streaming algorithm exists that (ε, δ)-approximates the number of simplices in $\mathcal{H}$ using $\widetilde{O}(m^{5/3} \log^2 n/S)$ space.*

5.1.3 Generalizing to k-graphs

Theorem 5.1.1 can be easily generalized to k-uniform hypergraphs. The idea of the algorithm remains the same: each simplex is labeled by an edge-vertex pair, where the vertices are ordered by degree. An edge is first sampled uniformly. If u is the vertex on that edge with the smallest degree, then then a vertex from the neighborhood of u is sampled. Using a final pass, we check whether the assembled label is a simplex and perform multiple iterations in order to bring the variance down. As before, primitives for ℓ_0 sampling and F_0 estimation are needed and are assumed to function perfectly.

The analysis does not change much. The estimator X is shown to be unbiased in precisely the same way. To bound the variance, we go through the same procedure to get

$$\mathbf{Var}[X] \leq (k-1) m^{3/2} S + m \sum_{e \in \mathcal{E}} S_e^2 \tag{5.17}$$

By Lemma 2.1.2, we see that $S_e \leq |S_u| = O(\sqrt{m})$, so the final bound on the variance is:

$$\boxed{\mathbf{Var}[X] = O(m^{3/2} S)} \tag{5.18}$$

Remark: *Note that the variance of the estimator in the general case is still $O(m^{3/2} S)$, due to the existence of the $m \sum_{e \in \mathcal{E}} S_e^2$ term.*

To figure out the expected memory usage of this algorithm, we use Theorem 3.2.4 to calculate:

$$\mathbf{Exp}[r] = \frac{1}{m} \sum_{\{u_1,\dots,u_k\}\in\mathcal{E}} \left\lceil \frac{\min\{\deg(u_1), \deg(u_2), \dots, \deg(u_k)\}}{\sqrt{m}} \right\rceil \tag{5.19}$$

$$= O\left(m^{-3/2} \sum_{\{u_1,\dots,u_k\}\in\mathcal{E}} \min\{\deg(u_1), \deg(u_2), \dots, \deg(u_k)\} \right) = O(m^{\frac{1}{2}-\frac{1}{k}}) \tag{5.20}$$

Algorithm 4: Counting simplices in k-uniform hypergraph streams

procedure $(k+1)$-SIMPLEX-STREAM-COUNT(σ)
  Pass 1 ▷ $O(1)$ space
    Pick an edge $\{u_1, u_2, \dots, u_k\} \in \mathcal{E}$ using reservoir sampling.
  Pass 2 ▷ $O(1)$ space
    Calculate the degrees $\deg(u_1), \deg(u_2), \dots, \deg(u_k)$.
  Pass 3 ▷ $O(S(F_0) + rS(L_0))$ space
    Let $r := \lceil \min\{\deg(u_1), \deg(u_2), \dots, \deg(u_k)\} \cdot m^{-1/2} \rceil$.
    Re-arrange (if needed) $u_1, u_2, \dots, u_k$ so that $u_1 \prec_\emptyset u_2 \prec_\emptyset \cdots \prec_\emptyset u_k$.
    for $j = 1$ to r **do**:
      $Z_j \leftarrow 0$
      Construct a virtual stream σ_j of the vertices in $N(u_1) \setminus \{u_2, \dots, u_k\}$.
      Sample vertex x_j from $N(u_1) \setminus \{u_2, \dots, u_k\}$ u.a.r using ℓ_0-sampling on σ_j
    end for
    Simultaneously, approximate $|N(u_1)|$ using F_0-estimation (▷ $O(\log n)$ space).
  Pass 4 ▷ $O(r)$ space
    Compute the degrees $\deg(x_1), \deg(x_2), \dots, \deg(x_r)$.
    for $j = 1$ to r **do**
      if $u_k \prec_\emptyset x_j$ **and** $\{u_1, u_2, \dots, u_k, x_j\}$ form a 4-simplex **then**
        $Z_j = |N(u_1)| - k + 1$ ▷ u_1 has the smallest degree
      end if
    end for
  $Y \leftarrow \frac{1}{r}\left(\sum_{j=1}^{r} Z_j\right)$ ▷ Average out the trials
  return $X = mY$ ▷ Scale to get an unbiased estimator
end procedure

Using the median-of-means improvement (2.2.1), we arrive to the following theorem:

Theorem 5.1.2. *If $\mathcal{H} = (V, \mathcal{E})$ is a k-uniform hypergraph with n vertices and m edges, then a 4-pass streaming algorithm exists that (ε, δ)-approximates the number of simplices in $\mathcal{H}$ using $\widetilde{O}(m^{2-\frac{1}{k}} \log^2 n/S)$ space.*

5.2 A codegree-based approach

A disadvantage of the algorithms described in the previous section is that they need to use ℓ_0 sampling and F_0 estimation as primitives. The underlying reason behind why they need to use such primitives is the difference between *neighborhood size* and *degree*. The neighborhood of a set of vertices is much harder to keep track of because many vertices can be incident on many edges simultaneously.

Another bothersome aspect of Algorithms 3 and 4 is their high memory requirements. As we will prove in Chapter 6, no algorithm can solve the simplex-counting problem in a 3-uniform hypergraph using $o(m^{4/3}/S)$ bits of space in the worst case. Clearly, there is a gap between the space requirements of Algorithm 3 and this lower bound. The inefficiency in memory usage seems to stem from two factors:

- **High variance**: $\mathbf{Var}[X] = O(m^{3/2} \cdot S)$. Our improved algorithm below will have $\mathbf{Var}[X] = O(m^{1+\frac{1}{k}} \cdot S)$ for $k \geq 3$.
- **Large memory usage per iteration**: In algorithm 4, $\mathbf{Exp}[B] \approx \mathbf{Exp}[r] = O(m^{\frac{1}{2}-\frac{1}{k}})$. We strive for $\mathbf{Exp}[B] = O(1)$.

An improved, efficient algorithm for simplex counting will need to address the issues above. First, we discuss one possible line of attack for improvement and show why it is not a fully efficient method.

5.2.1 Simplifying the sampling

A first way to improve Algorithm 4 by removing the ℓ_0 and F_0 primitives is to sample vertices adjacent to u_1 with probability proportional to the number of edges they share with u_1. This could be performed simply by using reservoir sampling on the virtual stream σ_j. The key observation to make is that the frequency of a vertex x in σ_j is exactly equal to its codegree $\deg_{G_{u_1}}(x)$. Since the vertices $u_2, ..., u_k$ are not in σ_j, we get by the generalized handshake lemma 2.1.1 for G_{u_1} that:

$$|\sigma_j| = (k-1)\deg(u_1) - \sum_{i=2}^{k} \deg_{G_{u_1}}(u_i) \tag{5.21}$$

This would sample a vertex $x \in N(u_1) \setminus \{u_2, ..., u_k\}$ with probability $\deg_{G_{u_1}}(x)/|\sigma_j|$. If V_E is the set of vertices that can complete a simplex label with edge E, and we assigned the value $|\sigma_j|/\deg_{G_{u_1}}(x)$ to the estimator X_j corresponding to sampling vertex x_j, then

$$\mathbf{Exp}[X_j|\mathcal{E}_E] = \sum_{z \in V_E} \left[\frac{\deg_{G_{u_1}}(z)}{|\sigma_j|} \cdot \frac{|\sigma_j|}{\deg_{G_{u_1}}(z)} \right] = |V_E| = S_E \tag{5.22}$$

So we can get an unbiased estimator and remove the ℓ_0/F_0 streaming primitives. In bounding the variance, we see that

$$\mathbf{Exp}[X_j^2 \mid \mathcal{E}_E] = \sum_{z \in V_E} \left[\frac{\deg_{G_{u_1}}(z)}{|\sigma_j|} \cdot \left(\frac{|\sigma_j|}{\deg_{G_{u_1}}(z)} \right)^2 \right] \leq |\sigma_j| \cdot \deg(u_1) \tag{5.23}$$

$$\mathbb{E}[X_{j_1} X_{j_2} \mid \mathcal{E}_E] = S_E^2 \tag{5.24}$$

Due to equation 5.24, the bound on the overall variance still depends on the additive term $\sum_{e \in \mathcal{E}} S_e^2$, which remains $O(m^{3/2})$ in our algorithm. Therefore, even though we removed a lot of complexity for our algorithm, the space complexity remains high due to the high variance.

5.2.2 An optimal algorithm for simplex counting

In order to lower the variance of our estimator, we will pursue techniques that work on the cases where the notions of degree and neighborhood size *coincide*. In k-uniform hypergraphs, this happens when examining the neighborhood and codegree of a set with $(k-1)$ vertices. This change in perspective leads us to the main idea that the algorithms to follow will use:

Main idea: Instead of looking at the neighbors of the min-degree vertex in a randomly chosen edge, we sample from the neighborhood of the $(k-1)$ vertices in the edge that have the smallest codegrees. To be more precise:

Definition 5.2.1. *If $e \in \mathcal{E}$ is some hyperedge, then we define:*

- *$c_1(e)$ to be the vertex of e with the smallest degree*
- *$c_i(e)$ is the vertex of e with the smallest $\{c_1(e), ..., c_{i-1}(e)\}$-codegree, for $1 < i < k$.*
- *Clearly, $c_k(e)$ is the vertex that remains.*

Our algorithms then sample from the neighborhood $N(c_1(e), ..., c_{k-1}(e))$ of a randomly chosen edge e. In the graph $G_{c_1(e),...,c_{k-1}(e)}$, the notions of neighborhood size and degree coincide, so there is no need for ℓ_0 sampling!

Using this idea, we arrive at an optimal algorithm for simplex counting:

Let $k \geq 2$. We associate a simplex S in $\mathcal{H}$ with a hyperedge-vertex pair (e, z), where:

- $S = e \cup \{z\}$
- For all $i = \{1, 2, ..., k-1\}$, if we let $S_i(e) := \{c_1(e), ..., c_i(e)\}$, we have that

$$c_i \prec_{S_{i-1}(e)} z$$

- $c_k(e) \prec_{S_{k-2}(e)} z$

We let T_e be the number of $(k+1)$-simplices labeled with edge e.

Observation 5.2.1. *For any simplex S there is a unique label (e, z) associated with S*

Proof. Let $S = \{u_1, ..., u_{k+1}\}$. Assume that two distinct labels (e_1, z_1) and (e_2, z_2) are associated with S. WLOG, assume that u_1 is the vertex with the smallest degree in S. Then u_1 must be in e_1 and e_2 by definition. Now, again WLOG, assume that u_2 is the vertex of

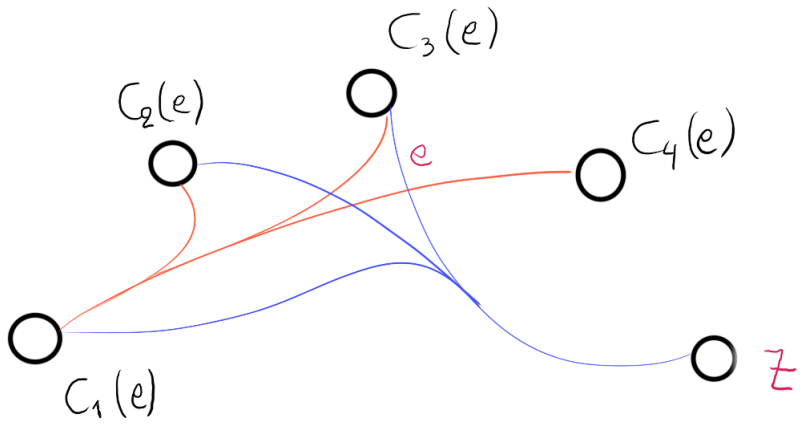

Figure 5.1: A simplex-characterization with $k = 4$.

$S \setminus \{u_1\}$ with the smallest u_1-codegree. u_2 must also belong to e_1 and e_2 by definition. We can repeat this process until $k-1$ vertices have been added to both e_1 and e_2. There are 2 vertices left: u_k and u_{k+1}. If we assume WLOG that u_k has smaller $\{u_1, ..., u_{k-2}\}$-codegree than u_{k+1}, then, by definition, u_k must be in e_1 and e_2. At that point we can see that $e_1 = e_2$ and so $z_1 = z_2$ - a contradiction. □

Our general algorithm follows an importance sampling approach: it samples an edge and a vertex and checks if the assembled label characterizes a simplex. Throughout the algorithm and the analysis, we treat k as a constant.

5.2.3 Analysis

We proceed in a fashion similar to Algorithms 3 and 4 above. We can show that our estimator is unbiased by using the same logic as we did before. If $\mathcal{E}_e$ is the event of sampling edge e in Line 3, then

$$\mathbb{E}[Z_k \,|\, \mathcal{E}_e] = T_e \frac{1}{\deg(S_{k-1}(e))} \deg(S_{k-1}(e)) = T_e \tag{5.25}$$

because $\deg(S_{k-1}(e)) = |N(S_{k-1}(e))|$. For the variance, we work in parallel to our previous discussion to establish, with the difference that now $\frac{\deg(S_{k-1}(e))}{r} = m^{1/k}$, which gives the following upper bound:

$$\mathrm{Var}[X] \leq m^{1+1/k} S + m \sum_{e \in \mathcal{E}} T_e^2 \tag{5.26}$$

The following lemma allows us to make this bound more compact:

Lemma 5.2.1. *For any $e \in \mathcal{E}$ we have that*

$$T_e = O(m^{1/k}) \tag{5.27}$$

Proof. We use induction on k. For $k = 2$ this is equivalent to Lemma 2.1.2. Now let $k \geq 3$. We distinguish between two cases:

Algorithm 5: Counting $(k+1)$-simplices in k-uniform hypergraph streams

procedure $(k+1)$-SIMPLEX-STREAM-COUNT(σ)
  Pass 1 ▷ $O(1)$ space
    Pick an edge $\{u_1, u_2, ..., u_k\} \in \mathcal{E}$ using reservoir sampling.
  Pass 2 ▷ $O(2^k) = O(1)$ space
    Calculate the degrees $\deg(S)$ for all non-trivial $S \subset e$.
  Pass 3 ▷ $O(r)$ space
    Re-arrange (if needed) $u_1, ..., u_k$ so that $u_i = c_i(e)$ for $i \in [k]$.
    With $S_j(e) := \{c_1(e), ..., c_j(e)\}$, let $r := \lceil \deg(S_{k-1}(e)) \cdot m^{-1/k} \rceil$.
    for $j = 1$ to r **do**:
      $Z_j \leftarrow 0$
      Construct a virtual stream σ_j consisting of the vertices in $N(S_{k-1}(e))$
      ▷ Note that $\deg(S_{k-1}(e)) = |N(S_{k-1}(e))|$
      Sample vertex x_j from $N(S_{k-1}(e))$ u.a.r using reservoir sampling.
    end for
  Pass 4 ▷ $O(kr) = O(r)$ space
    Compute the codegrees $\deg(S_{i-1}(e), x_j)$ for all $i \in [k]$ and $j \in [r]$
    for $j = 1$ to r **do**
      if $(c_i \prec_{S_{i-1}(e)} z,\ \forall i \in [k])$ and $(e \cup \{x_j\}$ form a 4-simplex$)$ **then**
        $Z_k = \deg(S_{k-1}(e))$
      end if
    end for
  return $X \leftarrow \frac{m}{r}\left(\sum_{k=1}^{r} Z_k\right)$ ▷ Average out the trials and scale
end procedure

1. $\mathbf{\deg(c_1(e)) \geq m^{(k-1)/k}}$. Then in any simplex labeled with edge e and vertex z, we must have that $\deg(z) \geq \deg(c_1(e))$. But there are at most $O(m^{1/k})$ such vertices due to the handshake lemma, so it follows that $T_e = O(m^{1/k})$.

2. $\mathbf{\deg(c_1(e)) < m^{k-1}/k}$. If $S = e \cup \{z\}$ is a simplex labeled with (e, z), then $S' = S \setminus \{c_1(e)\}$ must also be a simplex in $G_{S_1(e)}$. Further, S' must be labeled with $(e \setminus \{c_1(e)\}, z)$ in $G_{S_1(e)}$ because codegrees in $G_{S_1(e)}$ are $(c_1(e))$-codegrees in $\mathcal{H}$. So $T_e \leq T_{e\setminus\{c_1(e)\}}$.

   G_{S_1} is a $(k-1)$-uniform hypergraph and so, using our induction hypothesis, we have that $T_{e\setminus\{c_1(e)\}} = O(|\mathcal{E}_{S_1(e)}|^{1/(k-1)})$. But $|\mathcal{E}_{S_1(e)}| = \deg(c_1(e))$, so we get that

$$T_e = O\left(\deg(c_1(e))^{1/(k-1)}\right) = O(m^{1/k}) \tag{5.28}$$

□

Thus our variance is bounded as $\mathrm{Var}[X] = O(m^{\frac{k+1}{k}} S)$, and the median of means trick gives us an algorithm that uses 4 passes and $O(m^{\frac{k+1}{k}} B/S)$ bits of space, where $B = \Theta(r)$. Thus, as before, we need to bound r in expectation. We find that:

$$\mathbb{E}[r] = m^{-(k+1)/k} \sum_{e \in \mathcal{E}} \deg(S_{k-1}(e)) \tag{5.29}$$

The following theorem allows us to claim that $r = O(1)$ in expectation:

Theorem 5.2.2. *Let $k \geq 3$. If $\mathcal{H}$ is a k-uniform hypergraph, then:*

$$\sum_{e \in \mathcal{E}} deg(S_{k-1}(e)) = O(m^{\frac{k+1}{k}})$$

Proof. We use induction on k. For $k = 2$, this is given by 3.2.4. Let $k \geq 3$ below.

Consider partitioning the vertex set into "light" and "heavy" vertices as follows:

$$L := \{u \in V \mid \deg(u) < m^{(k-1)/k}\}$$

$$H := \{u \in V \mid \deg(u) \geq m^{(k-1)/k}\}$$

Let $\mathcal{M}_u$ be the set of hyperedges in which vertex u has the minimum degree:

$$\mathcal{M}_u := \{e \in \mathcal{E} \mid u = c_1(e)\}$$

Then $\mathcal{E} = \cup_{u \in V} \mathcal{M}_u$, so we can write:

$$\sum_{e \in \mathcal{E}} \deg(S_{k-1}(e)) = \sum_{u \in V} \sum_{e \in \mathcal{M}_u} \deg(S_{k-1}(e)) \tag{5.30}$$

$$= \sum_{u \in L} \sum_{e \in \mathcal{M}_u} \deg(S_{k-1}(e)) + \sum_{u \in H} \sum_{e \in \mathcal{M}_u} \deg(S_{k-1}(e)) \tag{5.31}$$

We first take care of the first term of the last summation, with $u \in L$. Each hyperedge $e \in \mathcal{M}_u$ corresponds to an edge $e' = e \setminus \{u\} \in \mathcal{E}_u$ in the $(k-1)$-uniform hypergraph G_u. Hyperedge e' has $k-1$ vertices and we have that $S_{k-2}(e') \cup \{u\} = S_{k-1}(e)$. Therefore, $\deg(S_{k-1}(e)) = \deg_{G_u}(S_{k-2}(e'))$ in G_u and we have:

$$\sum_{u \in L} \sum_{e \in \mathcal{M}_u} \deg(S_{k-1}(e)) \leq \sum_{u \in L} \sum_{e' \in \mathcal{E}_u} \deg_{G_u}(S_{k-2}(e')) \tag{5.32}$$

Because G_u is a $(k-1)$-uniform hypergraph, through the induction hypothesis, we get that

$$\sum_{e' \in \mathcal{E}_u} \deg_{G_u}(S_{k-2}(e')) = O(|\mathcal{E}_u|^{k/(k-1)}) = O(\deg(u)^{k/(k-1)}) \implies \tag{5.33}$$

$$\sum_{u \in L} \sum_{e' \in \mathcal{E}_u} \deg_{G_u}(S_{k-2}(e')) \leq \sum_{u \in L} \deg(u)^{k/(k-1)} \tag{5.34}$$

$$= \sum_{u \in L} \deg(u) \deg(u)^{\frac{1}{k-1}} \tag{5.35}$$

$$= O(m^{1/k}) \times km \tag{5.36}$$

$$= O(m^{(k+1)/k}) \tag{5.37}$$

Next, we want to bound the sum

$$S_H = \sum_{u \in H} \sum_{e \in \mathcal{M}_u} \deg(S_{k-1}(e))$$

Let M be the multiset consisting of the $(k-1)$-subsets of vertices whose degrees appear in S_H. If $P = \{u_1, ..., u_{k-1}\}$ is one subset in M, let f_P be its multiplicity. Also, let M' be the support of M. we write:

$$M = \bigcup_{P \in M'} P^{(f_P)} \tag{5.38}$$

We easily see that:

$$S_H = \sum_{P \in M} \deg(P) = \sum_{P \in M'} \deg(P) \cdot f_P \tag{5.39}$$

We claim that $f_P = O(m^{1/k})$ for all $P \in M$. For some hyperedge we have that $P = S_{k-1}(e)$. Let $z = e \setminus P$ be the vertex of e not in P. Then, by definition, $\deg(z) \geq \deg(c_1(e)) = \deg(u) \geq m^{(k-1)/k}$. So we have that:

$$f_P \leq |\{z \in V \mid \deg(z) > m^{(k-1)/k}\}| = O(m^{1/k})$$

by the handshake lemma. Using this bound on f_P and the handshake lemma we can now get:

$$\begin{aligned} S_H = \sum_{u \in H} \sum_{e \in \mathcal{M}_u} \deg(S_{k-1}(e)) &= \sum_{P \in M'} \deg(P) \cdot f_P && (5.40) \\ &\leq O(m^{1/k}) \sum_{P \in M'} \deg(P) && (5.41) \\ &\leq O(m^{1/k}) \cdot km = O(m^{\frac{k+1}{k}}) && (5.42) \end{aligned}$$

thus concluding the proof.

□

After applying the median-of-means improvement in a way similar to Theorem 5.1.1, we arrive to the following result:

Theorem 5.2.3. *If $\mathcal{H} = (V, \mathcal{E})$ is a k-uniform hypergraph with n vertices and m edges, then a 4-pass streaming algorithm exists that (ε, δ)-approximates the number of simplices in $\mathcal{H}$ using $O(m^{1+\frac{1}{k}}/S)$ space.*

Remark: *This algorithm, as we shall see in the next chapter, is optimal in its space complexity. It also does not require any pre-processing, degree oracles or ℓ_0/F_0 streaming primitives. On the other hand, one disadvantage of this algorithm is the $O(2^k)$-space factor in pass 2. Because we treat k as a constant, this is negligible, but if k and n are comparable asymptotically, the algorithm becomes prohibitively expensive.*

5.3 Importance-sampling and oracles revisited

The codegree approach adopted in the previous section can also be combined with the importance-sampling approach taken for Theorem 4.1.1. In this section, we design and analyse an algorithm to count simplices in 3-uniform hypergraphs using a codegree oracle. The algorithm is space-optimal, as it runs in $\widetilde{O}(m^{4/3}/S)$ bits of space in the worst case. It uses two main ideas:

- Given a vertex u, consider **wedges** $\{\{v,w\},\{v,z\}\}$ in G_u. When v is the common vertex in such a wedge, we say that the wedge is *centered at* v.

  A wedge $\{v,w,z\}$ centered in v in G_u is called $\prec_u$**-consistent** if

  $$v \prec_u w \text{ and } v \prec_u z$$

  Let W_u be the number of $\prec_u$-consistent wedges in $\mathcal{H}$.

- We implicitly construct a small set U that contains every 4-simplex 4 times.

  As done a few times already, we will call a vertex u **light** if $\deg(u) < m^{2/3}$. Otherwise we call u **heavy**. Let L and H denote the set of heavy and light vertices respectively. We can break up U as $U = U_L \cup U_H$, where

  $$U_L = \{(u,(v,w,z)_d) \mid u \in L \text{ and } \{v,w,z\} \text{ is a } \prec_u\text{-consistent wedge centered at } v\} \tag{5.43}$$

  $$U_H = \{(u,(v,w,z)_d \mid u \in H \text{ and } \{v,w,z\} \in \mathcal{E}\} \tag{5.44}$$

  Our algorithm samples from U uniformly by making use of the following quantities:

  - $L_4 = |U_L| = \sum_{u \in L} W_u$
  - $H_4 = |U_H| = |H| \cdot m$

Remark: *For this section, we assume that we have access to an oracle for the degree of any vertex u and the codegree of any pair of vertices (u,v). Calculating these quantities from scratch requires $\widetilde{O}(n^2)$ space and one additional pass.*

How is the sampling in line 10 performed? We use the 2 pass wedge sampling outlined in [MVV16]. That algorithm uses ℓ_0 and ℓ_2 sampling to sample a single consistent wedge. We will assume that those primitives run without error, and we will not take them into account in our final space complexity because they are logarithmic factors. Note that this increases the complexity of our algorithm because one has to remember that the sampling procedures may fail and so there is an additional component which weakens the quality of the overall estimate.

Algorithm 6: 4-Simplex Counting in the Arbitrary-Order Streaming model

procedure DEGREE-BASED-4-SIMPLEX-COUNT(σ)
  Offline: Calculate the sets L and H, as well as H_4 using the degree oracle.
  Pass 1 ▷ $O(n^2)$ space
    Using the codegree oracle, calculate W_u for every $u \in L$.
    Use this information to calculate L_4
  Sample a pair $(u_i, (v_i, w_i, z_i)_d)$**,** s **times in parallel, as follows:**
    $F \leftarrow$ Flip a coin with probability $\frac{L_4}{L_4+H_4}$.
    if $F =$ HEADS **then**
      Sample $u_i \in L$ with probability $\frac{W_u}{L_4}$.
      Passes 2-3: Sample a $\prec_u$ consistent wedge (v_i, w_i, z_i) uniformly from G_u.
    else
      Sample $v \in H$ with probability $\frac{1}{|H|}$.
      Passes 2-3: Sample some hyperedge $\{v_i, w_i, z_i\}$ with probability $\frac{1}{m}$
    end if
  Pass 4:
    Check if $\{u_i, v_i, w_i, z_i\}$ forms a 4-simplex for each $i \in [s]$
    Let $X_i = 1$ if that is the case, with $1 \leq i \leq s$. Otherwise set $X_i = 0$.
  Output: $\frac{1}{s}\sum_{i=1}^{s} X_i$.
end procedure

5.3.1 Analysis

The analysis here has very similar structure to the one of Algorithm 1.

Lemma 5.3.1. $W_u = O(deg(u)^{3/2})$

Proof. Let $v \neq u$. Consider the set S_v^u of the neighbours z of v in G_u such that $v \prec_u z$. By definition, $\deg_{G_u}(v) \geq |S_v^u|$, so we then have by the handshake lemma:

$$2\deg(u) \geq \sum_{z \in S_v^u} \deg_{G_u}(z) \geq |S_v^u| \cdot \deg_{G_u}(v) \geq |S_v^u|^2 \implies |S_v^u| \leq \sqrt{2} \cdot \deg(u)^{1/2} \tag{5.45}$$

Now, each edge $(v, w) \in E_u$ with $v \prec_u w$ can be in at most $|S_v^u|$ $\prec_u$-consistent wedges. Thus $W_u \leq \deg(u) \cdot \sqrt{2} \cdot \deg(u)^{1/2} = O(\deg(u)^{3/2})$ □

Lemma 5.3.2. $|U| = |U_L \cup U_H| = O(m^{4/3})$

Proof. We can write $|U|$ as:

$$|U| = |U_L| + |U_H| = L_4 + H_4 = \sum_{u \in L} W_u + |H| \cdot m \tag{5.46}$$

From lemma 5.3.1, we get that

$$\sum_{v\in L} W_u = O\left(\sum_{v\in L} \deg(v)^{3/2}\right) \tag{5.47}$$

$$= O\left(\sum_{v\in L} \deg(v)^{1/2}\deg(v)\right) \tag{5.48}$$

$$= O\left(m^{\frac{2}{3}\times\frac{1}{2}} \sum_{v\in L} \deg(v)\right) = O(m^{4/3}) \tag{5.49}$$

Due to the handshake lemma, there are at most $(3m)^{1/3}$ vertices in H, so $H_4 = O(m^{4/3})$. Adding up L_4 and H_4, we get the desired $O(m^{4/3})$ upper bound. □

Lemma 5.3.3. *Every 4-simplex is found 4 times in U as an unlabeled quadruple $\{u, v, w, z\}$*

Proof. If a 4-simplex $\{u, v, w, z\}$ contains i light vertices and $4 - i$ heavy vertices, with $0 \le i \le 3$, then it appears i times in U_L and $4 - i$ times in U_H, for a total of 4 times. □

Lemma 5.3.4. *Our algorithm 6 samples a labeled quadruple uniformly from U.*

Proof. If $(u, \{v, w, z\})_d \in U_L$, then it is sampled with probability

$$\frac{\cancel{L_4}}{L_4 + H_4} \cdot \frac{\cancel{W_u}}{\cancel{L_4}} \cdot \frac{1}{\cancel{W_u}} = \frac{1}{L_4 + H_4} = \frac{1}{|U|} \tag{5.50}$$

If $(u, \{v, w, z\}) \in U_H$, then it is sampled with probability

$$\frac{\cancel{H_4}}{L_4 + H_4} \cdot \frac{1}{\cancel{|H|}} \cdot \frac{1}{\cancel{m}} = \frac{1}{L_4 + H_4} = \frac{1}{|U|} \tag{5.51}$$

□

Lemma 5.3.5. *If $s = \Theta\left(\frac{|U|\log n}{\varepsilon^2 S}\right)$, then Algorithm 6 (ε, n^{-d})-approximates S, where d is some constant greater than 1.*

Proof. First, it is easy to see that $\mathbb{E}[X_i] = \frac{4S}{|U|}$. Now, applying a standard Chernoff bound:

$$P\left\{\left|\frac{1}{s}\sum_{i=1}^{s} X_i - \frac{4S}{|U|}\right| \ge \varepsilon \cdot \frac{4S}{|U|}\right\} \le 2\exp\left(-\frac{4\varepsilon^2 Ss}{2|U|}\right) \le n^{-d} \tag{5.52}$$

□

Lemma 5.3.6. *Algorithm 6 uses $\widetilde{O}\left(n^2 + \frac{m^{4/3}}{S}\right)$ bits of space in the worst case.*

Proof. The sampling in lines 10 or 13 requires $\widetilde{\Theta}(1)$ space. Since $|U| = O(m^{4/3})$ (Lemma 5.3.2) and $s = \widetilde{\Theta}\left(\frac{|U|}{S}\right)$, we have that lines 6-18 require $O\left(\frac{m^{4/3}}{S}\right)$ bits of space.

□

Remark: *This algorithm could be generalized to count simplices for arbitrary k, but it looks like that would require $\Theta(k)$ passes and involve a $O(n^{k-1})$ additive constant to the space complexity. Due to these disadvantages, we are not explicitly exploring the generalization to an arbitrary k. On the other hand, we also have not explored whether some of these disadvantages can be removed.*

5.4 Other algorithms for simplex counting

To conclude this chapter, we present for the sake of completeness, two additional generalizations of well-known triangle counting algorithms to the hypergraph problem.

5.4.1 A reduction based algorithm

The paper [BYKS02] of Bar-Yossef et al (2002) is a seminal paper whose central goal is to introduce a general methodology for streaming algorithms: streaming reductions. By reducing problems of streaming nature to already-existing algorithms, we obtain efficient solutions. The paper introduced this idea abstractly and also developed some primitives for calculating frequency moments F_k. The reduction of triangle counting to frequency moment estimation is a truly clever algorithm and one of the first ones to tackle and solve this problem in the streaming setting.

It is straightforward to adapt their technique to hypergraph streaming. We will omit the space-complexity analysis, but will present the general methodology. Suppose we are trying to count simplices on $(k-1)$-uniform hypergraphs. On arrival of edge $e = \{u_1, u_2, ..., u_{k-1}\}$ we make the virtual stream consisting of k-tuples $e \cup \{v\} \neq e$, where $v \in V$. Then the p-th frequency moment of this virtual stream is:

$$F_p(\sigma) = \sum_{i=1}^{k} P_i i^p \tag{5.53}$$

where P_i is the number of k-tuples where exactly i hyperedges exist. Plugging in $p \in \{0, 1, 2, ..., k-1\}$, we get the system:

$$\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ . \\ . \\ . \\ F_{k-1} \end{bmatrix} = \begin{bmatrix} 1 & . & . & . & 1 \\ 1 & 2 & 3 & . & k \\ 1 & 4 & . & . & k^2 \\ 1 & 8 & . & . & k^3 \\ . & . & . & . & . \\ . & . & . & . & . \\ 1 & 2^{k-1} & . & . & k^{k-1} \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ . \\ . \\ . \\ P_k \end{bmatrix} \tag{5.54}$$

Solving for $[P_1 \cdots P_k]^T$ we just have to invert that matrix. This matrix is the inverse of a **Vandermonde matrix**, so its determinant is:

$$D = \prod_{1 \leq ij \leq k} (j - i) = (k-1)! \times (k-2)! \times \cdots \times 2 \times 1 \neq 0 \tag{5.55}$$

So we have to run algorithms for F_p estimation for $p = 0, 1, ..., k-1$. This is a rather inconvenient strategy because F_p-norm estimation cannot be done efficiently for $p \notin (0, 2)$.

What if we wanted to count K_k^r with $r < k$? We insert with each edge $e = (u_1, ..., u_r)$ the remaining $k-r$ vertices. This creates a stream with $m\binom{n-r}{k-r}$ elements. If P_i is the number of k tuples containing i edges with r vertices, then:

$$F_p(\sigma) = \sum_{i=1}^{\binom{k}{r}} P_i i^p \tag{5.56}$$

Using the same exact technique, this time our system has a Vandermonde matrix of size $\binom{k}{r} \times \binom{k}{r}$, but its determinant is still non-zero. Therefore the system can be solved and we can retrieve the desired value $P_{\binom{k}{r}}$.

5.4.2 Another sampling approach

In [BFL+06], Buriol et al introduced sampling techniques to the problem of triangle counting in streams. The main idea behind their algorithm was to pick an edge at random $e = (a, b)$ and a vertex $v \in V \setminus \{a, b\}$. The estimator takes the value 1 if $\{a, b, v\}$ forms a triangle and 0 otherwise. Repeating this around $O(\epsilon^{-2} \log \delta^{-1} mn/T)$ times and taking the average leads to an accurate and precise estimator estimator X with expected value $E[X] = \frac{3S}{m(n-2)}$. This algorithm is simple to implement and has an elegant analysis.

To count the occurrences of a k-clique K_k^r in a r-uniform hypergraph ($r < k$), this method could be generalized to collect an edge $e = \{u_1, ..., u_r\}$ and vertices $u_{r+1}, ..., u_k$. Then we can check on a second pass if the k-tuple we collected forms K_k^r. Let X be the random variable representing that outcome. Then,

$$E[X] = \frac{\binom{k}{r} T_k^r}{m\binom{n-r}{k-r}} \tag{5.57}$$

where T_k^r is the number of K_k^r copies in our hypergraph.

Picking an edge happens through reservoir sampling and picking $k - r$ vertices requires $O((k-r) \log n)$ memory. The space required for an average of the above estimators to be efficient and accurate is: $O(\varepsilon^{-2} \log \delta^{-1} m\binom{n-r}{k-r}/T_k^r)$. If $k = r = \Omega(n)$, this translates into very little memory required, as expected.

Chapter 6

Lower Bounds for Simplex Counting

This chapter concerns lower bounds for the problem of simplex counting in hypergraph streams. By proving a lower bound for the worst-case memory usage of any constant-pass streaming algorithm, we finally complete the proof of Theorem 1.3.1.

We will be using methods from communication complexity to reach the desired results. For more information on prerequisite knowledge about Communication Complexity, please refer to Chapter 2. The results in this chapter will all be lower bounds of the form: *"No algorithm can solve problem ____ using $o(f(n, m, \varepsilon, k))$ bits of memory in the worst case"*, where f is some function of the number of vertices, edges, uniformity and precision in a simplex counting problem.

6.1 General approach

Given a stream σ describing a k-uniform hypergraph $\mathcal{H} = (V, \mathcal{E})$ with n vertices and m edges, we want to produce an estimate of S - the number of $(k+1)$-simplices in $\mathcal{H}$. If we denote our estimate by $\widetilde{S}$, then we want $\widetilde{S} \in [(1-\varepsilon)S, (1+\varepsilon)S]$ with probability at least $2/3$. Here we have $\varepsilon \in (0, 1)$ is a constant which characterizes the precision of our estimate. Any algorithm that can achieve such an approximation using at most p passes over σ solves the problem $\textbf{Simplex-Count}_k(n, m, \varepsilon)$.

If $T \geq 1$ is some constant threshold and we just want to decide whether $\mathcal{H}$ contains 0 or at least T $(k+1)$-simplices with probability at least $2/3$ in p passes, then we want to solve the problem $\textbf{Simplex-Dist}_{\mathbf{k}}(\mathbf{n}, \mathbf{m}, \mathbf{T})$. Note that we could have that $0 < S < T$, but we do not care in this case what the algorithm returns. We just want it to work right when $S = 0$ or $S \geq T$. In some sense, this a a *promise problem*.

If $T \geq 1$ is again some constant threshold and we wish to decide whether $S > (1+\varepsilon)T$ or $S \leq (1-\varepsilon)T$ with probability at least $2/3$ in p passes over σ, then we are solving the promise problem $\textbf{Simplex-Sep}_{\mathbf{k}}(\mathbf{n}, \mathbf{m}, \mathbf{T}, \varepsilon)$. As before, if $S \in ((1-\varepsilon)T, (1+\varepsilon)T]$, we do not care what our algorithm returns.

Intuitively, Simplex-Count$_k$ is harder to solve than Simplex-Dist$_k$ or Simplex-Sep$_k$, because a good approximation to S allows us to easily decide about a general range S lies in. This is made formal below:

Theorem 6.1.1. Simplex-Dist$_k$ *and* Simplex-Sep$_k$ *reduce to* Simplex-Count$_k$ *with no additional memory.*

Proof. Let $\mathcal{A}$ be an algorithm solving Simplex-Count$_k(n, m, \varepsilon)$.

For Simplex-Dist$_k(n, m, T)$, we just run $\mathcal{A}$ and check if the output $\widetilde{S}$ is 0 or at least $(1-\varepsilon)T$. If $S = 0$, then $\widetilde{S}$ must also be zero with probability at least 2/3. If $S \geq T$, then with probability at least 2/3 we will know that $(1-\varepsilon)T \leq (1-\varepsilon)S \leq \widetilde{S}$, so we answer correctly.

For Simplex-Sep$_k(n, m, T, \varepsilon)$, we will run $\mathcal{A}$ are check if the output $\widetilde{S}$ is greater than $(1-\varepsilon^2)T$. If so we will decide that $S > (1+\varepsilon)T$. Otherwise we will decide that $S \leq (1-\varepsilon)T$. There are again two cases (because we do not care what happens if $S \in ((1-\varepsilon)T, (1+\varepsilon)T]$):

- $S > (1+\varepsilon)T$. Then our verdict will be wrong if $\widetilde{S} \leq (1-\varepsilon^2)T$. But $\widetilde{S} \geq (1-\varepsilon)S > (1-\varepsilon^2)T$ with probability at least 2/3. Therefore we will be wrong with probability at most 1/3.
- $S < (1-\varepsilon)T$. With probability at least 2/3 we know that $\widetilde{S} \leq (1+\varepsilon)S \leq (1-\varepsilon^2)T$, so we are right again.

Both of these reductions require no additional space, so we conclude the proof. □

In the following sections we will deduce theorems about the hardness of the Simplex-Dist and Simplex-Sep problems. By the preceding Theorem 6.1.1 this will also imply theorems about the hardness of Simplex-Count.

6.2 A lower bound in terms of n

If we parametrize the space complexity just in terms of the number of vertices n, then we get a discouraging lower bound result.

Suppose we have the problem of counting all the copies of K_k^r, the r-uniform hypergraph with k vertices in an r-uniform hypergraph $\mathcal{H}$ with n vertices and m edges. We provide a lower bound to a much simpler, yet equally hard problem: distinguishing whether $\mathcal{H}$ contains zero or more copies of K_k^r. For K_3^2 (triangles), the construction is classic: we reduce from the *disjointness problem* in communication complexity. We split $3n$ vertices into 3 parts and make a construction in which Alice and Bob's inputs share a common 1 if and only if the construction has a triangle.

We could try generalizing this approach to K_k^{k-1} first. Alice and Bob have inputs $A, B \in \{0,1\}^{n^{k-1}}$. Assume of course that $k \leq n$. They view their inputs as $(k-1)$-dimensional cubes with side length n. As they parse their input, they parse the edges of an implicit hypergraph $\mathcal{H}$: The vertices of $\mathcal{H}$ are split into 3 groups: $A = \{a_1, ..., a_n\}, B = \{b_1, ..., b_n\}, C = \{c_1, ..., c_n\}$. If $A[i_1, i_2, ..., i_{k-1}] = 1$, then Alice connects with a hyperedge the vertices a_{i_1} and $\{c_{i_2}, c_{i_3}, c_{i_4}, ..., c_{i_{k-1}}\}$. When she is done constructing this stream, she passes her information to Bob, who does the same, except he includes vertex b_{i_1} instead of a_{i_1}. Bob also includes in his stream the $n\binom{n}{k-3}$ hyperedges of the form $\{a_i, b_i\} \cup C'$ for all $(k-3)$-sized subsets of C.

Lemma 6.2.1. *DISJ*$(A, B) = 1$ *if and only if* G *contains a copy of* K_k^{k-1}.

Proof. If A and B have a common 1 in places $(i_1, i_2, ..., i_{k-1})$, then the edges $\{a_{i_1}, c_{i_2}, ..., c_{i_{k-1}}\}$, $\{b_{i_1}, c_{i_2}, ..., c_{i_{k-1}}\}$ and all the edges containing a_{i_1}, b_{i_1} and the $(k-3)$-sized subsets of $C'' = \{c_{i_2}, c_{i_3}, ..., c_{i_{k-1}}\}$ are contained in $\mathcal{H}$. This is k hyperedges in total, forming a simplex. Conversely, if G contains a copy of K_k^{k-1}, then the vertices of that copy must be some set of the form $\{a_{i_1}, b_{i_1}, c_{i_2}, ..., c_{i_{k-1}}\}$ because of the way we set up $\mathcal{H}$. So that means that $A[i_1, i_2, ..., i_{k-1}] = B[i_1, i_2, ..., i_{k-1}] = 1$. □

Since $\text{DISJ}_{n^{k-1}}$ requires $\Omega(n^{k-1})$ bits of communication in the worst case to even approximate the right answer with some probability of error, our algorithm for simplex distinguishing also requires $\Omega(n^{k-1})$ bits of space in the worst case. Therefore, we have proven the following theorem:

Theorem 6.2.1. *Any (possibly randomized) streaming algorithm that distinguishes between a hypergraph having zero or more copies of* K_k^{k-1} *requires* $\Omega(n^{k-1})$ *space in the worst case. If we allow* p *passes over the stream, then this lower bound becomes* $\Omega\left(\frac{n^{k-1}}{2p-1}\right)$.

What about counting K_k^r? The generalization is easy: But this time Alice has to add $\binom{k-2}{r-1}$ hyperedges for every element. Bob, along with his edges, also adds edges of the form $\{a_i, b_i\} \cup C'$, where $C' \subseteq C$ and $|C'| = r - 2$. The overall lower bound remains $\Omega(n^{k-1})$.

Theorem 6.2.2. *Any (possibly randomized) streaming algorithm that distinguishes between a hypergraph having zero or more copies of* K_k^r *requires* $\Omega(n^{k-1})$ *space in the worst case.*

6.3 Lower bounds dependent on ε

In this section we present some hardness results for the space complexity of streaming algorithms solving Simplex-Count in 3-uniform hypergraphs. *We parameterize the space complexity in terms only of the precision parameter* ε.

Our results are stated as impossibility theorems rather than lower bounds to the space complexity. We prove that no algorithm can achieve a specified upper bound on all inputs by examining certain hard instances. This is different from proving Ω-style lower bounds, as that type of result would have to hold for all possible inputs, whereas we simply exhibit hard instances.

6.3.1 $o(\varepsilon^{-1})$ is impossible

Theorem 6.3.1. *There is no algorithm solving* Simplex-Sep$_3$ *with precision parameter* ε *using* $o(\varepsilon^{-1})$ *bits of space in the worst case.*

Proof. Assume that n is the smallest positive integer which is such that $\frac{1}{n+1} \leq \varepsilon \leq \frac{1}{n}$. We first prove the impossibility result for $\varepsilon = \frac{1}{n}$.

We reduce from the 3-party communication problem U-DISJ$_n$. Alice, Bob and Charlie hold strings $x, y, z \in \{0,1\}^n$ respectively. We think of x, y, z as the characteristic sets of subsets $X, Y, Z \subseteq [n]$. We operate under the promise that $X \cap Y \cap Z = \emptyset$ or $|X \cap Y \cap Z| = 1$.

We use a p-pass streaming algorithm $\mathcal{A}$ that solves SIMPLEX-SEP with parameter $\frac{1}{n}$ to design a p-round protocol for U-DISJ$_n$. To this end, we transform an instance (x, y, z) of U-DISJ$_n$ to an instance of SIMPLEX-SEP$_3$ - that is to a 3-uniform hypergraph $\mathcal{H}$ and a parameter T.

This is how we construct $\mathcal{H} = (\mathbf{V}, \mathcal{E})$ and implement a protocol reduction for DISJ$_n$:

1. $V = \{a_1, ..., a_n\} \cup \{b_1, ..., b_n\} \cup \{c_1, ..., c_n\} \cup \{d_1, ..., d_n\} \cup \{q_1, ..., q_n\} \cup \{r_1, ..., r_n\}$. Clearly $|V| = 6n = N$. Alice starts running $\mathcal{A}$ on the following hyperedges:

2. Alice first adds the following $4n^3 - n^2$ "static" hyperedges:

   - $\{\{a_i, b_j, c_k\}, \{a_i, b_j, d_k\}, \{a_i, c_j, d_k\} \mid (i,j,k) \in [n]^3\}$
   - $\{\{b_i, c_j, d_k\} \mid (i,j,k) \in [n]^3 \text{ and } k \neq 1\}$. She excludes the n^2 (b, c, d_1) hyperedges.

3. For every index $k \in [n^2]$ such that $x[k] = 1$, Alice adds the n^2 hyperedges

   $$\{\{a_i, b_j, q_k\}, \{a_i, b_j, r_k\} \mid (i,j) \in [n]^2\}$$

   In the worst case she adds n^3 hyperedges. Alice then copies the memory tape of $\mathcal{A}$ at this point onto the shared whiteboard.

4. Bob retrieves the memory tape from the whiteboard and continues running A with n^2 hyperedges for each $k \in [n]$ such that $y[k] = 1$:

   $$\{\{a_i, c_j, q_k\}, \{a_i, c_j, r_k\} \mid (i,j) \in [n]^2\}$$

   In the worst case he adds n^3 hyperedges. In the end, he copies down the memory tape of $\mathcal{A}$ onto the shared whiteboard.

5. Finally, Charlie retrieves the memory tape and continues running A with the edges:

   $$\{\{b_i, c_j, q_k\}, \{b_i, c_j, r_k\} \mid (i,j) \in [n]^2\}$$

   where $z[k] = 1$. In the worst case he adds n^3 hyperedges.

6. **Let $\mathbf{T} = \mathbf{n^4}$.** If this is not the p-th pass over the stream, Charlie writes down the memory tape contents onto the whiteboard and we go back to step 2. Otherwise Charlie outputs:

   - $|X \cap Y \cap Z| = 1$, if $\mathcal{A}$ returns "$S > (1 + \frac{1}{n})T$ ".
   - $|X \cap Y \cap Z| = 0$, if $\mathcal{A}$ returns "$S < (1 - \frac{1}{n})T$ ".

During the protocol above, we constructed a hypergraph $\mathcal{H}$ and streamed it to $\mathcal{A}$. $\mathcal{H}$ has

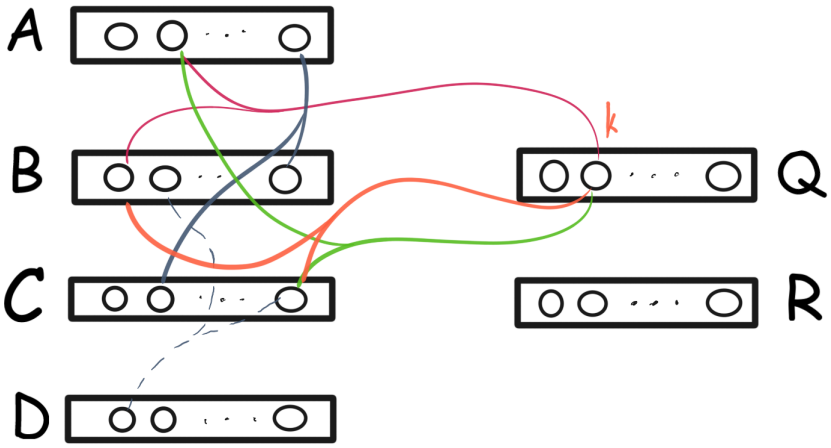

Figure 6.1: The reduction of Theorem 6.3.1. The dotted hyperedge is missing. $2n^3$ simplices are created with an intersection.

$6n$ vertices and $m \leq 4n^3 - n^2 + 6n^3 = \Theta(n^3)$ hyperedges. The following lemma allows us to complete our reduction:

Lemma 6.3.1. *$\mathcal{H}$ has more than $(1 + \frac{1}{n})T$ 4-simplices if and only if $|X \cap Y \cap Z| = 1$. Otherwise $\mathcal{H}$ has at most $(1 - \frac{1}{n})T$ 4-simplices.*

Proof. Alice's "static" edges make up $n^4 - n^3 = (1 - \frac{1}{n})T$ 4-simplices. Assume that $X \cap Y \cap Z = \{l\}$. Then every quadruple $\{a_i, b_j, c_k, q_l\}$ and $\{a_i, b_j, c_k, r_l\}$ is a 4-simplex, so there are at least $n^4 - n^3 + 2n^3 = (1 + \frac{1}{n})T$ 4-simplices. On the other hand, if $|X \cap Y \cap Z| = 0$, there are no additional simplices. □

To conclude our proof, we recall that U-DISJ$_n$ requires $\Omega(n)$ bits of communication. However, this is also a lower bound on the bits of space $\mathcal{A}$ must use. If $\mathcal{A}$ uses SPACE($\mathcal{A}$) bits of space then after p passes the protocol uses $(3p - 1)$SPACE(A) bits of communication. Therefore, since $p = O(1)$ we have SPACE(A) $= \Omega(n)$.

We now generalize to an arbitrary ε. Since $\varepsilon \geq \frac{1}{n}$, an algorithm solving SIMPLEX-SEP$_3$ with parameter ε can also solve SIMPLEX-SEP$_3$ with parameter $\frac{1}{n}$. Since $n \leq \varepsilon^{-1} \leq n + 1$, we see than no algorithm can solve SIMPLEX-SEP$_3$ in $o(\varepsilon^{-1})$ space. □

Theorem 6.3.2. *There is no algorithm solving* SIMPLEX-COUNT$_3$ *with precision parameter ε using $o(\varepsilon^{-1})$ bits of space in the worst case.*

Proof. We just combine Theorem 6.3.1 with Lemma 6.1.1. □

6.3.2 $o(\varepsilon^{-2})$ is impossible

We now prove a slightly stronger theorem by utilizing a different communication problem, which is outlined in Chapter 2.

Theorem 6.3.3. *There is no algorithm solving* Simplex-Sep$_3$ *with precision parameter* ε *using* $o(\varepsilon^{-2})$ *bits of space in the worst case.*

Proof. Similarly to Theorem 6.3.1, we assume that $\frac{2}{\sqrt{n}+1} \leq \varepsilon \leq \frac{2}{\sqrt{n}}$ and show the impossibility result for $\varepsilon = \frac{2}{\sqrt{n}}$. We skip the generalization details for arbitrary ε as they are identical to Theorem 6.3.1.

We will reduce from the Gap-Hamming$_3$ problem in the multi-party, number-in-hand, blackboard communication model.

Alice, Bob and Charlie hold strings $x, y, z \in \{\pm 1\}^n$. They are *promised* that the Gap-Hamming distance $D(x, y, z)$ of their strings satisfies either $D(x, y, z) \geq n/2 + \sqrt{n}$ or $D(x, y, z) \leq n/2 + \sqrt{n}$ and they want to figure out which is the case.

Let $\mathcal{A}$ be a $p = O(1)$-pass streaming algorithm for solving Simplex-Sep$_3$ with parameter $\frac{2}{\sqrt{n}}$. Alice, Bob and Charlie want to construct a hypergraph $\mathcal{H} = (V, \mathcal{E})$ based on (x, y, z) and input this hypergraph as a stream to $\mathcal{A}$ along with some parameter T. The parties communicate $\mathcal{A}$'s memory tape by writing it on the blackboard every time their turn has ended. This way they make a communication protocol that solves Gap-Hamming$_3$. Because $R_\varepsilon(\text{Gap-Hamming}_3) = \Omega(n)$ and $p = O(1)$, $\mathcal{A}$ must require $\Omega(n)$ bits of memory.
We set $\mathbf{T = n^4/2}$. The construction of $\mathcal{H} = (\mathbf{V}, \mathcal{E})$ is as follows:

- $V = \{a_1, ..., a_n\} \cup \{b_1, ..., b_n\} \cup \{c_1, ..., c_n\} \cup \{s_1, ..., s_n\} \cup \{t_1, ..., t_n\}$.
- Alice first adds all n^3 edges of the form $\{\{a_i, b_j, c_k\} \mid i, j, k \in [n]\}$.
- If $x[k] = 1$ for some $k \in [n]$, Alice adds the n^2 edges of the form $\{a_i, b_j, s_k\}$. If $x[k] = -1$, she instead adds $\{a_i, b_j, t_k\}$ edges.
- If $y[k] = 1$ for some $k \in [n]$, Bob adds the n^2 edges of the form $\{b_i, c_j, s_k\}$. If $y[k] = -1$, he instead adds $\{b_i, c_j, t_k\}$ edges.
- If $z[k] = 1$ for some $k \in [n]$, Charlie adds the n^2 edges of the form $\{a_i, c_j, s_k\}$. If $z[k] = -1$, he instead adds $\{a_i, c_j, t_k\}$ edges.
- Eventually, Charlie says "$D(x, y, z) \geq n/2 + \sqrt{n}$ " if $\mathcal{A}$ says "$T_3 \leq (1 + \frac{2}{\sqrt{n}})T$ ".

The above construction shows that $\mathcal{H}$ has $N = 5n$ vertices and $m = \Theta(n^3)$ edges. We now need to show that solving Simplex-Sep$_3$; on $\mathcal{H}$ with parameter $\frac{2}{\sqrt{n}}$ solves Gap-Hamming$_3$:

Lemma 6.3.2. $\mathcal{H}$ *has more than* $(1+\frac{2}{\sqrt{n}})T$ 4*-simplices if and only if* $D(x, y, z) \leq n/2-\sqrt{n}$. *Otherwise* $\mathcal{H}$ *has at most* $(1 - \frac{2}{\sqrt{n}})T$ 4*-simplices.*

Proof. 4-simplices exist in $\mathcal{H}$ only due to positions of matching characters in x, y and z. If $D(x, y, z) \leq n/2 - \sqrt{n}$, then there are *at least* $n/2 + \sqrt{n}$ positions $i \in [n]$ such that $x[i] = y[i] = z[i]$. So, there are at least $(n/2 + \sqrt{n})n^3 = \frac{n^4}{2} + n^{7/2} = (1 + \frac{2}{\sqrt{n}})T$ 4-simplices of the form $\{a_i, b_j, c_k, s_r\}$ or $\{a_i, b_j, c_k, t_r\}$. Conversely, if $D(x, y, z) \geq n/2 + \sqrt{n}$, there are at most $\frac{n^4}{2} - n^{7/2} = (1 - \frac{2}{\sqrt{n}})T$ 4-simplices. □

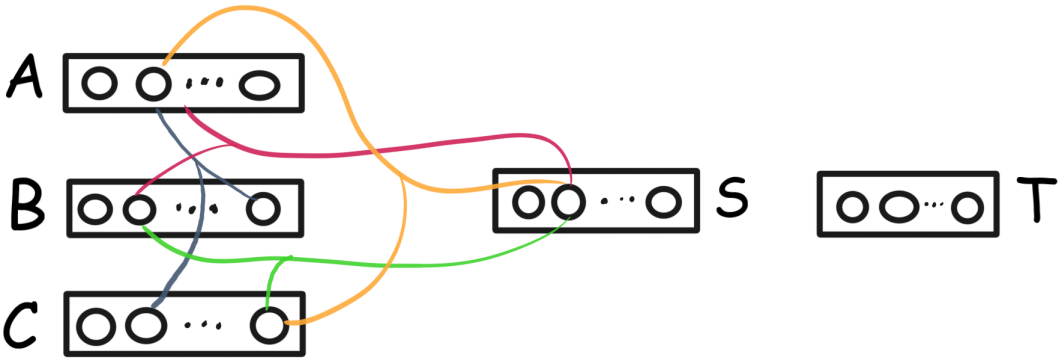

Figure 6.2: The $o(\varepsilon^{-2})$ construction showing an example of a matching position in x, y, z.

Consequently, $\mathcal{A}$ cannot solve Simplex-Sep$_3$ using $o(n)$ space. Because $n \leq \varepsilon^{-2} \leq n+1$, we get that no algorithm can solve Simplex-Sep$_3$ with parameter ε using $o(\varepsilon^{-2})$ bits of space. □

From Theorem 6.3.3 and Theorem 6.1.1, we can arrive at the following impossibility result:

Theorem 6.3.4. *There is no algorithm solving* Simplex-Count$_3$ *with precision parameter ε using $o(\varepsilon^{-2})$ bits of space in the worst case.*

6.4 Lower Bounds Dependent on m and S

In this section we present hardness results for the space complexity of streaming algorithms solving Simplex-Count in uniform hypergraphs, under the promise that $S \geq T$ for some parameter T. We parameterize the space complexity in terms of the number of edges m and the number of simplices S.

6.4.1 Weaker results

$o(m/S)$ is impossible

We first present a sub-optimal hardness result for the Simplex-Dist$_3$ problem in terms of m and the parameter T. We then use the promise $S \geq T$ and Theorem 6.1.1 to derive the $o(m/S)$ impossibility for Simplex-Count$_3$.

Theorem 6.4.1. *No algorithm solving* Simplex-Dist$_3$ *with parameter T on a hypergraph $\mathcal{H}$ with m hyperedges can use $o(m/T)$ bits of space in the worst case.*

Proof. The idea behind this reduction is almost identical to the $\Omega(n^{k-1})$ lower bound we proved earlier. Let n be a sufficiently large positive integer. Let $p = n^3$ and $r = \frac{n^3}{3}$. We reduce from the Disj$_p^{r,T}$ communication problem, where $T \leq r$.

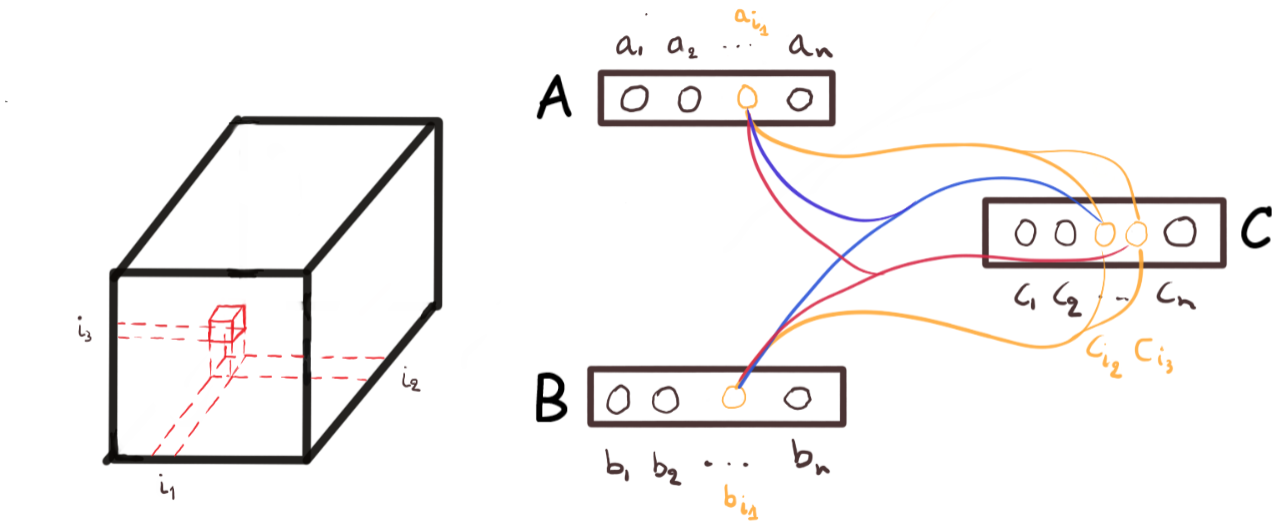

Figure 6.3: The reduction of Theorem 6.4.1 portrayed in a sketch

Alice and Bob hold strings $x, y \in \{0,1\}^{n^3}$. We view x and y as 3-dimensional adjacency matrices. We have three sets of vertices $A = \{a_1, ..., a_n\}$, $B = \{b_1, ..., b_n\}$ and $C = \{c_1, ..., c_n\}$ and if $x[i_1, i_2, i_3] = 1$, Alice inserts the hyperedge $\{a_{i_1}, c_{i_2}, c_{i_3}\}$. Bob does the same with y, but instead he inserts the hyperedge $\{b_{i_1}, c_{i_2}, c_{i_3}\}$. He also inserts the edges $\bigcup_{i=1}^{n} \{\{a_i, b_i, c_j\} \mid c_j \in C\}$ at the end.

Our constructed hypergraph $\mathcal{H}$ has at least T 4-simplices if and only if $x[i_1, i_2, i_3] = y[i_1, i_2, i_3] = 1$ at least T distinct times. The number of edges m in $\mathcal{H}$ is $m = \frac{2n^3}{3} + n^2 = \Theta(r)$. If some constant-pass streaming algorithm solved Simplex-Dist$_3$ using $o(m/T)$ bits of space in the worst case, then then Alice and Bob would be able to communicate $\text{DISJ}_p^{r,T}$ using $o(m/T) = o(r/T)$ communication, which contradicts the $\Omega(r/T)$ lower bound on the communication complexity of this function. □

Theorem 6.4.2. *No algorithm solving* Simplex-Count$_3$ *on hypergraphs* $\mathcal{H}$ *with* m *hyperedges and* S 4*-simplices under the promise that* $S \geq T$ *can use* $o(m/S)$ *bits of space in the worst case.*

Proof. By Theorem 6.1.1, such an algorithm could solve Simplex-Dist$_3$ with parameter T using $o(m/S) = o(m/T)$ space, which contradicts Theorem 6.4.1 □

$o(m/S^{3/4})$ is impossible

We can strengthen our previous hardness result even further:

Theorem 6.4.3. *No algorithm solving* Simplex-Dist$_3$ *with parameter T on a hypergraph $\mathcal{H}$ with m hyperedges can use $o(m/T^{3/4})$ bits of space in the worst case.*

Proof. Let $\mathcal{A}$ be an algorithm solving Simplex-Dist$_3$ with parameter T, using $o(m/T^{3/4})$ bits of space in the worst case. Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$ with m edges and S simplices, we make a hypergraph $\mathcal{H}' = (V', \mathcal{E}')$ by

- Replacing each vertex $v \in V$ with T vertices $v_1, ..., v_T$
- Replacing each hyperedge $\{a, b, c\}$ with the T^3 hyperedges $\{a_i, b_j, c_k\}$.

Any 4-simplex in $\mathcal{H}$ is now replaced with T^4 simplices in $\mathcal{H}'$. Furthermore, any simplex in $\mathcal{H}'$ corresponds to a simplex in $\mathcal{H}$. So $\mathcal{H}'$ has $T^3 m$ edges and $T^4 S$ simplices.

We can now reduce from the problem Simplex-Dist$_3(T = 1)$: $\mathcal{H}$ has at least one 4-simplex if and only if $\mathcal{H}'$ has at least T^4 4-simplices. $\mathcal{A}$ uses $o(mT^3/(T^4)^{-3/4}) = o(m)$ bits of space on $\mathcal{H}'$, thus solving Simplex-Dist$_3(T = 1)$ in $o(m)$ space, which is impossible by Theorem 6.4.1. Therefore, $\mathcal{A}$ cannot use $o(m/T^{3/4})$ space in the worst case. □

As before, the promise $S \geq T$ gives us the following impossibility result for Simplex-Count$_3$:

Theorem 6.4.4. *No algorithm solving* Simplex-Count$_3$ *on hypergraphs $\mathcal{H}$ with m hyperedges and S 4-simplices under the promise that $S \geq T$ can use $o(m/S^{3/4})$ bits of space in the worst case.*

6.4.2 Optimal lower bounds

The constructions in this section are very similar conceptually to the $o(\varepsilon^{-2})$ construction made above. Please refer to Figure 6.2 for an illustration of those ideas.

$o(m^{4/3}/S)$ is impossible

Theorem 6.4.5. *No algorithm solving* Simplex-Dist$_3$ *with parameter T on a hypergraph $\mathcal{H}$ with m hyperedges can use $o(m^{4/3}/T)$ bits of space in the worst case.*

Proof. We reduce from the Disj communication problem. Alice and Bob hold n-bit strings $x, y \in \{0, 1\}^n$ respectively. These strings represent subsets of $[n]$. Alice and Bob want to determine whether the represented sets have an intersection (Disj$(x, y) = 0$) or not. We know from communication complexity theory that any randomized protocol that allows Alice and Bob to answer correctly with probability at least 2/3 has to use $\Omega(n)$ bits of communication.

Let $\mathcal{A}$ be an algorithm for solving Simplex-Dist$_3$. Alice uses her x to construct a hypergraph stream and feed it into $\mathcal{A}$. The constructed hypergraph $\mathcal{H}$ has $4n$ vertices split into 4 groups of n: $A = [a_i]_{i=1}^n, B = [b_i]_{i=1}^n, C = [c_i]_{i=1}^n$ and $D = [d_i]_{i=1}^n$. If $x[i] = 1$ for some $i \in [n]$, Alice feeds n^2 hyperedges of the form $\{d_i, a_j, b_k\}$ into $\mathcal{A}$.

After she finishes parsing x, she sends to Bob the memory tape of $\mathcal{A}$ and Bob continues running $\mathcal{A}$ by parsing through his y the same way Alice did. He adds $2n^2$ hyperedges when $y[i] = 1$: the hyperedges $\{d_i, b_j, c_k\}$ and $\{d_i, a_j, c_k\}$. After Bob finishes parsing y he adds n^3 hyperedges of the form $\{a_i, b_j, c_k\}$ to $\mathcal{A}$. If, after that, $\mathcal{A}$ responds that $\mathcal{H}$ has at least $T = n^3$ simplices then Bob decides that $\textsc{Disj}(x, y) = 0$.

In the end, $\mathcal{H}$ has $\Theta(n^3)$ edges. Further, x and y have a non-empty intersection if and only if $\mathcal{H}$ has at least n^3 simplices. Now, if $\mathcal{A}$ used $o(m^{4/3}/T)$ bits of memory, this communication protocol would use $o(n^4/n^3) = o(n)$ bits of memory, which contradicts the known lower bound for the communication complexity of Disj.

□

Theorem 6.4.6. *No algorithm solving* Simplex-Count$_3$ *on hypergraphs* $\mathcal{H}$ *under the promise that* $S \geq T$ *can use* $o(m^{4/3}/S)$ *bits of space in the worst case.*

Proof. By Theorem 6.1.1, such an algorithm could solve Simplex-Dist$_3$ with parameter T using $o(m^{4/3}/S) = o(m^{4/3}/T)$ space, which contradicts Theorem 6.4.5 □

Generalizing to k-graphs

Theorem 6.4.7. *No algorithm solving* Simplex-Dist$_k$ *with parameter* T *on a hypergraph* $\mathcal{H}$ *with* m *hyperedges can use* $o(m^{\frac{k+1}{k}}/T)$ *bits of space in the worst case.*

Proof. As before, we reduce from the Disj communication problem. Alice and Bob hold n-bit strings $x, y \in \{0,1\}^n$ respectively.

Let $\mathcal{A}$ be an algorithm for solving Simplex-Dist$_k$. Alice uses her x to construct a hypergraph stream and feed it into $\mathcal{A}$. The constructed hypergraph $\mathcal{H}$ has $(k+1)n$ vertices split into $(k+1)$ groups of n: $A_j = [a_i^j]_{i=1}^n$ for $j \in [k]$ and $D = [d_i]_{i=1}^n$. If $x[i] = 1$ for some $i \in [n]$, Alice feeds n^{k-1} hyperedges of the form $\{d_i, a_{i_1}^1, a_{i_2}^2, ..., a_{i_{k-1}}^{(k-1)}\}$ into $\mathcal{A}$.

After she finishes parsing x, she sends to Bob the memory tape of $\mathcal{A}$ and Bob continues running $\mathcal{A}$ by parsing through his y the same way Alice did. He adds $(k-1)n^{k-1}$ hyperedges when $y[i] = 1$: the hyperedges $\left\{d_i, a_{i_1}^1, ..., a_{i_{j-1}}^{j-1}, a_{i_{j+1}}^{j+1}, ..., a_{j_k}^k\right\}$ for $j = 1, 2, ..., (k-1)$. After Bob finishes parsing y he adds n^k hyperedges of the form $\{a_{i_1}^1, ..., a_{i_k}^k\}$ to $\mathcal{A}$. If, after that, $\mathcal{A}$ responds that $\mathcal{H}$ has at least $T = n^k$ simplices then Bob decides that $\textsc{Disj}(x, y) = 0$.

In the end, $\mathcal{H}$ has $\Theta(n^k)$ edges. Further, x and y have a non-embpty intersection if and only if $\mathcal{H}$ has at least n^k simplices. Now, if $\mathcal{A}$ used $o(m^{\frac{k+1}{k}}/T)$ bits of memory, this communication protocol would use $o(n^{k+1}/n^k) = o(n)$ bits of memory, which contradicts the known lower bound for the communication complexity of Disj.

□

Theorem 6.4.8. *No algorithm solving* Simplex-Count$_k$ *on hypergraphs* $\mathcal{H}$ *with* m

hyperedges and S $(k+1)$-simplices under the promise that $S \geq T$ can use $o(m^{\frac{k+1}{k}}/S)$ bits of space in the worst case.

Proof. By Theorem 6.1.1, such an algorithm could solve Simplex-Dist$_3$ with parameter T using $o(m^{\frac{k+1}{k}}/S) = o(m^{\frac{k+1}{k}}/T)$ space, which contradicts Theorem 6.4.5 □

Chapter 7

Conclusion

In this thesis, the problem of counting the number of simplices in uniform hypergraph streams was explored from multiple perspectives. An efficient k-simplex algorithm was given in the arbitrary-order streaming model for when $k \ll n$. A matching lower bound was also proven in terms of multiple different space parameterization options, effectively resolving the most important open question regarding the space complexity of the problem.

Further, the problem of triangle counting and sampling was studied, and space-optimal streaming algorithms were developed and analyzed for both problems. These algorithms have not appeared in the literature, and may be of value due to their simplicity for implementation purposes.

Regarding implementation, a necessary future follow up work to this thesis is the implementation of the algorithms developed. By implementing and testing them against real-life data, we can determine their practical advantages and shortcomings, which will motivate future research questions. For instance, if the assumption that $k \ll n$ is removed, the space requirements of Algorithm 5 become prohibitively large. A natural empirical question is whether such instances exist in real-life hypergraph networks, and if so, is there a way to remove the 2^k-factor in the space complexity?

A related question has to do with lower bounds. In terms of k, how much memory does an algorithm that solves the simplex question require? Such a lower bound was not investigated in this thesis, and would settle the question of whether the 2^k factor can be removed or whether it is part of what makes the problem difficult.

Overall, one may be surprised by the fact that the optimal algorithm for simplex counting requires $\widetilde{O}\left(\frac{m^{1+\frac{1}{k}}}{T}\right)$ bits of space in the worst case. The surprise factor lies in the fact that the space goes to $O(m/T)$ as $k \to \infty$, independently of n. Therefore, it gets *easier and easier* to count simplices as their complexity increases. Why is that? An intuitive answer to this question may be that more complex simplices are *fewer* in number in a hypergraph. Therefore, an importance sampling technique that reduces the search space to a sufficiently small number of vertex k-tuples should be able to estimate S without using much memory.

Our formalized approach to this intuitive idea is through the *codegree characterization* of the vertices in each hyperedge. By restricting the available candidates for completing a simplex to an extreme amount, we reduce the variance of our estimator to an optimal amount. Besides that, the idea of *degree-based vertex partitioning* is the other crucial tool our algorithms use.

A few other follow-up questions may be raised at this point[1]. First, our algorithm is sub-optimal when the hypergraphs contain few simplices (T is small). Indeed, one could simply store all the edges in that case and that would save space. To remedy this situation for triangles, a $O\left(\frac{m}{\sqrt{T}}\right)$ algorithm has been given, and thus the complexity of the triangle counting problem is typically written as $O\left(\min\left\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\right\}\right)$. It is very interesting to ask whether a similar approach can be discovered for hypergraphs as well.

Also, our algorithm uses 4 passes over the input stream. One has to wonder if this is the best possible number of passes required to achieve this space complexity. Indeed, it might be the case that a 1-pass algorithm, for instance, requires more memory usage, so we have to use multiple passes. Such a result would come about possibly from a reduction from the INDEX problem in communication complexity, and will definitely be the focus of future work.

Finally, the problems explored in this thesis open up more lines of research in field of pattern counting in streamed graphs and hypergraphs. One could ask about counting or enumerating other types of patterns, like cycles, star-like shapes and motifs of various kinds. Research could also explore the area of directed hypergraphs (or *dihypergraphs*) and study how the problems change in that setting. Further, the *uniformity assumption* could be removed. The hypergraphs we study are all uniform, but what if that was not the case? Is counting patterns in non-uniform hypergraphs as hard or possibly easier than the case where uniformity is given? Further research can definitely shed a lot of light on such questions about hypergraphs in the streaming setting.

[1]Thank you Professors Deeparnab Chakrabarty and Hsien-Chih Chang for motivating these questions

Bibliography

[BC17] Suman K. Bera and Amit Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In Heribert Vollmer and Brigitte Vallée, editors, *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[BDGL08] Ilaria Bordino, Debora Donato, Aristides Gionis, and Stefano Leonardi. Mining large networks with subgraph counting. In *2008 Eighth IEEE International Conference on Data Mining*, pages 737–742. IEEE, 2008.

[BFKP16] Laurent Bulteau, Vincent Froese, Konstantin Kutzkov, and Rasmus Pagh. Triangle counting in dynamic graph streams. *Algorithmica*, 76(1):259–278, 2016.

[BFL+06] Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 253–262, 2006.

[BOV13] Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. How hard is counting triangles in the streaming model? In *International Colloquium on Automata, Languages, and Programming*, pages 244–254. Springer, 2013.

[BR+91] Richard A Brualdi, Herbert J Ryser, et al. *Combinatorial matrix theory*, volume 39. Springer, 1991.

[Bre13] Alain Bretto. Hypergraph theory: An introduction. *Mathematical Engineering. Cham: Springer*, 2013.

[Bru10] Richard A Brualdi. Spectra of digraphs. *Linear Algebra and its Applications*, 432(9):2181–2213, 2010.

[BS20] Suman K Bera and C Seshadhri. How the degeneracy helps for triangle counting in graph streams. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 457–467, 2020.

[BYKS02] Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, volume 2, pages 623–632, 2002.

[CF10] Nicholas A Christakis and James H Fowler. Social network sensors for early detection of contagious outbreaks. *PloS one*, 5(9):e12948, 2010.

[CJ17] Graham Cormode and Hossein Jowhari. A second look at counting triangles in graph streams (corrected). *Theoretical Computer Science*, 683:22–30, 2017.

[CJLT12] Yeow Meng Chee, Lijun Ji, Andrew Lim, and Anthony KH Tung. Arboricity: An acyclic hypergraph decomposition problem motivated by database theory. *Discrete applied mathematics*, 160(1-2):100–107, 2012.

[CKS03] Amit Chakrabarti, Subhash Khot, and Xiaodong Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *18th IEEE Annual Conference on Computational Complexity, 2003. Proceedings.*, pages 107–117. IEEE, 2003.

[CMW+94] Boliong Chen, Makoto Matsumoto, Jianfang Wang, Zhongfu Zhang, and Jianxun Zhang. A short proof of Nash-Williams' theorem for the arboricity of a graph. *Graphs and Combinatorics*, 10(1):27–28, 1994.

[CN85] N. Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14:210–223, 1985.

[ELRS15] Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. USA, 2015. IEEE Computer Society.

[FG06] Jörg Flum and Martin Grohe. *Parameterized complexity theory*. Springer Science & Business Media, 2006.

[FH97] Ioannis Fudos and Christoph M Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics (TOG)*, 16(2):179–216, 1997.

[FKK03] András Frank, Tamás Király, and Matthias Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Applied Mathematics*, 131(2):373 – 383, 2003. Submodularity.

[GMT15] Sudipto Guha, Andrew McGregor, and David Tench. Vertex and Hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 241–247, 2015.

[Gow06] W Timothy Gowers. Quasirandomness, counting and regularity for 3-uniform hypergraphs. *Comb. Probab. Comput.*, 15(1-2):143–184, 2006.

[Hae95] Willem H Haemers. Interlacing eigenvalues and graphs. *Linear Algebra and its applications*, 226:593–616, 1995.

[Her20] Matthias Hermann. Graph sparsification techniques for triangle counting. Master's thesis, 2020.

[HW07] Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

[JLL+20] Tanqiu Jiang, Yi Li, Honghao Lin, Yisong Ruan, and David P Woodruff. Learning-augmented data stream algorithms. *ICLR*, 2020.

[JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58, 2011.

[JW21] Rajesh Jayaram and David Woodruff. Perfect l_p sampling in a data stream. *SIAM Journal on Computing*, 50(2):382–439, 2021.

[Kee11] Peter Keevash. Hypergraph Turan problems. *Surveys in combinatorics*, 392:83–140, 2011.

[KKP18] John Kallaugher, Michael Kapralov, and Eric Price. The sketching complexity of graph and hypergraph counting. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 556–567. IEEE, 2018.

[KMPT12] Mihail N Kolountzakis, Gary L Miller, Richard Peng, and Charalampos E Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.

[KN96] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, USA, 1996.

[KNPW11] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 745–754, 2011.

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 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 '10, page 41–52, New York, NY, USA, 2010. Association for Computing Machinery.

[LHK10] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650, 2010.

[Lov68] László Lovász. On chromatic number of finite set-systems. *Acta Mathematica Academiae Scientiarum Hungarica*, 19(1-2):59–67, 1968.

[Lov70] László Lovász. A generalization of Kónig's theorem. *Acta Mathematica Academiae Scientiarum Hungarica*, 21(3-4):443–446, 1970.

[MV20] Andrew McGregor and Sofya Vorotnikova. Triangle and four cycle counting in the data stream model. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 445–456, 2020.

[MVV16] Andrew McGregor, Sofya Vorotnikova, and Hoa T Vu. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 401–411, 2016.

[PRS05] Yuejian Peng, Vojtech Rödl, and Jozef Skokan. Counting small cliques in 3-uniform hypergraphs. *Combinatorics, Probability and Computing*, 14(3):371–413, 2005.

[PT12] Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a MapReduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.

[PTTW13] A. Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Counting and sampling triangles from a graph stream. *Proc. VLDB Endow.*, 6(14):1870–1881, September 2013.

[RY20] Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020.

[SJHS15] Benjamin Schiller, Sven Jager, Kay Hamacher, and Thorsten Strufe. StreaM - a Stream-Based Algorithm for Counting Motifs in Dynamic Graphs. pages 53–67, 08 2015.

[SOK+20] Kijung Shin, Sejoon Oh, Jisu Kim, Bryan Hooi, and Christos Faloutsos. Fast, accurate and provable triangle counting in fully dynamic graph streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(2):1–39, 2020.

[Sun13] He Sun. Counting hypergraphs in data streams. *arXiv preprint arXiv:1304.7456*, 2013.

[TKMF09] Charalampos E Tsourakakis, U Kang, Gary L Miller, and Christos Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 837–846, 2009.

[Vor20] Sofya Vorotnikova. Improved 3-pass algorithm for counting 4-cycles in arbitrary order streaming. *arXiv preprint arXiv:2007.13466*, 2020.

[WCW+18] Zhiang Wu, Jie Cao, Yaqiong Wang, Youquan Wang, Lu Zhang, and Junjie Wu. hPSD: A Hybrid pu-learning-based Spammer Detection Model for Product Reviews. *IEEE transactions on cybernetics*, 50(4):1595–1606, 2018.

[Yus06] Raphael Yuster. Finding and counting cliques and independent sets in r-uniform hypergraphs. *Information Processing Letters*, 99(4):130–134, 2006.