Stochastic Streams:
Sample Complexity vs. Space Complexity

Michael Crouch¹, Andrew McGregor*,¹, and David Woodruff²

¹University of Massachusetts at Amherst
²IBM Almaden

Abstract

We address the trade-off between the computational resources needed to process a large
data set and the number of samples available from the data set. Specifically, we consider
the following abstraction: we receive a potentially infinite stream of IID samples from some
unknown distribution $D$, and are tasked with computing some function $f(D)$. If the stream
is observed for time $t$, how much memory, $s$, is required to estimate $f(D)$? We refer to $t$
as the sample complexity and $s$ as the space complexity. The main focus of this paper is
investigating the trade-offs between the space and sample complexity. We study these trade-offs
for two canonical problems: undirected graph connectivity and estimating frequency moments.
Our algorithms are based on techniques for emulating random walks and simulating different
sampling procedures given a sequence of IID samples.

1 Introduction

If you want to understand the world around you, would you rather live a long life with a poor
memory, or a short life with a good memory? Or, on a more technical level, what is the trade-off
between the computational resources needed to process a large data set and the number of samples
available from the data set. We consider these questions in terms of the following abstraction: we
receive a stream of IID samples from some unknown distribution $D$, and are tasked with estimating
some function $f(D)$ of the distribution. Two natural questions arise, both of which have been
studied extensively:

1. The statistics question is how to bound the sample complexity: how many samples are required
to estimate $f(D)$ to some prescribed accuracy with high probability?

2. The data stream question is how to bound the space complexity: how much memory is
required to compute or approximate the estimator for $f(D)$?

*Supported by NSF CAREER Award CCF-0953754.
The focus of this paper is investigating the trade-offs between these two quantities: If the stream is observed for time \( t \), how much memory, \( s \), is required to estimate \( f(D) \). Obviously \( t \) needs to be at least the sample complexity \( t^* \) for the problem since, with less than \( t^* \) samples, insufficient information has been revealed about the data set and therefore even unlimited computation space and time cannot yield a good answer. However, in this paper we will observe that for two canonical problems, the space complexity to process these samples decreases as \( t/t^* \) increases.

**Sufficient Statistics and Data Streams.** The goal of space-efficiency in statistical estimation is not new. In the study of **sufficient statistics** [2] the goal is to prove that it suffices to maintain a small number of statistics about the input when estimating certain parameters of the source distribution \( D \). For example, to estimate \( \mu \) if \( D \sim N(\mu, 1) \), it is sufficient to maintain the sum and count of the samples; other information can be discarded. However, for non-parametric problems sufficient statistics typically do not exist. Therefore, it makes sense to also consider “approximate sufficient statistics”, i.e., statistics about the stream of samples that can be computed online that will suffice to estimate the relevant properties of the input with high probability.

In contrast to the majority of data stream research, in the above setting we do not need to consider adversarially-ordered streams since the assumption is that input stream is generated by a stochastic process. There is a growing body of work on randomly-ordered streams [3–9]. Some work has also explicitly considered streams of IID samples [10–13]. There has also been work on hypothesis testing given limited space [14, 15].

**Subsampling vs. Supersampling.** Other related work includes a paper by an overlapping set of authors [16] (see also [17]) that considered the problem of processing data streams whose arrival rate was so high that it was not possible to observe every element of the stream. Consequently, it was presumed that the stream was first **subsampled** and then properties of the original stream had to be deduced from the samples. In contrast, in this work we essentially consider oversampling, or **supersampling** the data set. The motivation is two-fold. First, in many applications there is an abundance of redundant data, e.g., from sensors that a continually monitoring a static environment, and it makes sense to find a way to capitalize on this data. Second, it may be preferable from a computational point of view, to run a fast light-weight algorithm on a lot of data rather than a computationally expensive algorithm on a small amount of data.

**Our Results.**

We study the trade-off between sample and space complexity for the canonical problems of undirected connectivity [18–20] and estimating frequency moments [21, 22]. Our results are as follows:

1. **Frequency Moments.** Suppose \( D = D_p \) is a distribution \((p_1, \ldots, p_n)\) over \([n]\). Then, to estimate \( F_k(D) = \sum_i p_i^k \) for \( k \geq 2 \) up to a factor \((1 + \epsilon)\) it is sufficient\(^\dagger\) for

   \[
   s \cdot t = \tilde{O}_\epsilon(n^{2-2/k})
   \]

   if \( t \geq t^* = \Omega(n^{1-1/k}) \). We also show the lower bound that \( s = \tilde{\Omega}(n^{2.5(1-1/k)}/t^{1.5}) \) space is necessary to return a constant approximation. We present the algorithmic result in Section 2 and the lower bound in Section 4.

\(^\dagger\tilde{O}_\epsilon(f(n))\) omits \( \text{poly}(\log n, 1/\epsilon) \) terms.
2. **Graph Connectivity.** Suppose $D = D_G$ is the uniform distribution of the set of edges of an undirected, unweighted graph $G$. We show that it suffices for
\[ s^2 \cdot t = \tilde{O}(|E| \cdot |V|^2) \]
if $t \geq t^* = \Omega(|E| \log |E|)$. We present these results in Section 3.

2 **Frequency Moments**

In this section we consider a stream of samples $\langle a_1, a_2, a_3, \ldots \rangle$ where each $a_i$ is drawn independently from some unknown, discrete distribution $p$ over $[n]$. Let $p_j = \Pr [a_i = j]$ and define $F_k = \sum_j p_j^k$. Our main theorem in this section is as follows:

**Theorem 2.1.** For $k \geq 2$ and for $t = \Omega(n^{1-1/k})$, estimating $F_k$ up to a factor $(1 \pm \epsilon)$ given $t$ samples is possible in $\tilde{O}(n^{2-2/k}/t)$ space.

We will prove the upper bound in this section and establish a lower bound in Section 4. We start with a simple algorithm for the $k = 2$ case and then extend to $k > 2$ via a variant of $\ell_2$ sampling. Note that the condition that $t = \Omega(n^{1-1/k})$ was shown to be necessary by Bar-Yossef [23]; with less samples it is information-theoretically impossible to estimate $F_k$.

2.1 **Warm-Up: $F_2$ Estimation**

Consider the stream of samples as defining a sequence of pairs
\[ \langle (a_1, a_2), (a_3, a_4), \ldots, (a_{t-1}, a_t) \rangle. \]

Note that the probability that a pair of values is identical is exactly $F_2$. Define $X_i \in \{0, 1\}$ where $X_i = 1[a_{2i-1} = a_{2i}]$. Let $X = \frac{2}{t} \sum_i X_i$.

**Lemma 2.2.** If $t > 6e^{-2}n \ln(2/\delta)$ then $\Pr [\|X - F_2\| \geq \epsilon F_2] \leq \delta$.

**Proof.** First note that $\mathbb{E} [X] = \sum_i p_i^2 = F_2$. Since each $X_i \in \{0, 1\}$ is independent, by an application of the Chernoff bound, $\Pr [\|X - F_2\| \geq \epsilon F_2] \leq 2e^{-\epsilon^2 F_2/6}$. Hence this probability is less than $\delta$ if $t \geq 6 \ln(2/\delta)/(F_2 \epsilon^2)$. Since $F_2 \geq 1/n$ the result follows.

2.2 **Technique: $\ell_2$-Sampling in the IID model**

The main technique that will be used in our $F_k$ estimation algorithm is a form of $\ell_2$ sampling. Here we are given a sequence of samples $\langle a_1, a_2, a_3 \ldots \rangle$ from $p$ and wish to generate a sample from $q$ where $q_i = p_i^2/F_2(p)$. In the $F_2$-estimation algorithm above, we observed that returning $a_{2i}$ for the smallest $i$ such that $a_{2i-1} = a_{2i}$ yields a value drawn according to $q$. This required $O(\log n)$ space but potentially $\Omega(n)$ samples even in expectation.

We next consider a generalization of this idea: rather than taking two samples and hoping they are the same value, we take $w$ samples and return the first value that occurs twice in the sequence. If $w$ is small, it is possible there will be no duplicates. However, the next lemma establishes that $i$ is sampled with probability approximately proportional to $p_i^2$. 
Lemma 2.3. Probability of returning \( i \in [n] \) satisfies:

\[
\frac{cp_i^2}{1 + wp_i} \leq \mathbb{P}[\text{returning } i] \leq cp_i^2
\]

where \( c \) does not depend on \( i \). The probability of returning any value in \([n]\) is at least \( \min(1/2, w^2/(50n)) \).

Proof. To simplify notation let \( i = n \). Let

\[
A_k = \sum_{\text{distinct } i_1, \ldots, i_k \in [n]} p_{i_1} \ldots p_{i_k} \quad \text{and} \quad B_k = \sum_{\text{distinct } i_1, \ldots, i_k \in [n-1]} p_{i_1} \ldots p_{i_k}
\]

Then, the probability of returning \( n \) can be written as \( p_n^2 (1 + 2B_1 + 3B_2 + \ldots + (w - 1)B_{w-2}) \).

Since \( A_k = B_k + kp_nB_{k-1} \), we then have

\[
1 + 2B_1 + 3B_2 + \ldots + (w - 1)B_{w-2} \\
\leq 1 + 2A_1 + 3A_2 + \ldots + (w - 1)A_{w-2} \\
= (1 + 2p_n) + 2(1 + 3p_n)B_1 + 3(1 + 4p_n)B_2 + \ldots \\
+ (w - 2)(1 + (w - 1)p_n)B_{w-3} + (w - 1)B_{w-2}
\]

The first part of the lemma follows by setting \( c = 1 + 2A_1 + 3A_2 + \ldots + (w - 1)A_{w-2} \).

For the second part, observe that the probability of returning a value in \([n]\) is minimized if \( p \) is the uniform distribution over \([n]\). With probability at least \( 1/2 \) there is at least one pair of duplicate samples after \( 5\sqrt{n} \) samples. If such a pair exists, the probability that a specific pair occurs within the first \( w \) samples is \( \min(1, (w/5\sqrt{n})^2) \) since every ordering of the \( 5\sqrt{n} \) samples is equally likely. \( \square \)

2.2.1 Frequency Moments Algorithm and Analysis

The algorithm we present is inspired by the \( \ell_2 \)-sampling approach for frequency moment estimation \([24,25]\). Specifically, suppose we had a sequence of \( r = O(\epsilon^{-2}n^{1-2/k} \log \delta^{-1}) \) samples and frequency estimations \((i_1, \tilde{p}_{i_1}), (i_2, \tilde{p}_{i_2}), \ldots (i_r, \tilde{p}_{i_r})\), where \( \tilde{p}_{i_j} = (1 \pm \epsilon/2)p_{i_j} \) and where samples were chosen with probability \( \mathbb{P}[i_j = \ell] = (1 \pm \epsilon/2)p_{i_j}^2/F_2(p) \). Then a simple calculation and an application of the Chernoff bound would allow us to estimate \( F_k(p) \) via

\[
\Pr\left[ \left| \frac{F_2(p)}{r} \sum_{j=1}^{r} \tilde{p}_{i_j}^{k-2} - F_k(p) \right| > \epsilon F_k(p) \right] \leq \delta .
\]

We can estimate \( F_2(p) \) in the sampling model as above. The remaining challenge is to approximate \( \sum_{j=1}^{r} \tilde{p}_{i_j}^{k-2} \). We present an algorithm which approximates this by sampling \( r \) elements (with appropriate probabilities) and then approximating their frequencies.

In what follows we assume all \( p_i \leq \epsilon^3 t/n^{2-2/k} \) since we can identify all larger probabilities (there will be only \( O(\epsilon^{-3}n^{2-2/k}/t) \) of them and they’ll all appear among the first \( O(\epsilon^{-3} \log nn^{2-2/k}/t) \) samples) and estimate their frequencies separately.
Algorithm. We consider the \( t \) samples divided into \( t/w \) contiguous segments 

\[ C_1, C_2, \ldots, C_{t/w}, \]

each consisting of \( w = \max\{5\sqrt{n}, \alpha \epsilon^{-2} \log \delta^{-1}/n^{1-2/k}/t\} \) samples for some large constant \( \alpha \). In each segment \( C_i \), the goal is to collect the first duplicate \( e_i \). Given the assumption that \( p_i \leq \epsilon/w \) for all \( i \), Lemma 2.3 ensures that the element is chosen with probability \( \propto (1 \pm \epsilon)p_i^2 \) as required. Subsequently we monitor \( e_i \), perhaps into future segments, until we have a good estimate of \( p_{e_i} \).

1. \( X \leftarrow 0 \)
2. Collect: For each segment \( C_i \), let \( e_i \) be the first element to have a duplicate (if one exists).
3. Monitoring: Let \( c_i \) be the number of samples in \( C_{i+1} \cup C_{i+2} \cup \ldots \) that appear before \( e_i \) has occurred \( d = O(\epsilon^{-2} \log n) \) times. Let \( X \leftarrow X + (d/c_i)^{k-2}/r \)
4. Repeat until \( r \) samples \( e_i \) are found. Return \( F_2 \cdot X \).

The following lemma establishes that we get a good approximation of the probability of each element collected. It follows immediately from the tail bounds for the negative binomial distribution (see Lemma A.1 in the appendix).

**Lemma 2.4.** For all \( i \in [r] \), \( d/c_i = (1 \pm \epsilon/2)p_i \)

**Lemma 2.5.** With high probability, the algorithm terminates before the end of the stream is reached.

**Proof.** By Lemma 2.3, the probability of finding a duplicate during a segment is at least \( \min(1/2, w^2/(50n)) \). Hence the expected number of elements collected is 

\[ \min(1/2, w^2/(50n)) \cdot t/w = tw/(50n) \geq \alpha \epsilon^{-2} \log \delta^{-1}/n^{1-2/k}/50. \]

For sufficiently large \( \alpha > 0 \), this is at least \( r \) with high probability.

The last lemma bounds the space use of the algorithm.

**Lemma 2.6.** The algorithm uses \( \tilde{O}_\epsilon(n^{2-2/k}/t) \) space.

**Proof.** To find a duplicate in each segment only requires \( w \leq n^{2-2/k}/t \) space. It remains to show that we are never monitoring too many items at the same time. During the \( j \)-th segment, we may be estimating \( p_i \). By Lemma A.1 we may assume that we have not been monitoring \( i \) for more than \( O(\epsilon^{-2} \log n/p_i) \) time, or equivalently more than \( O(\epsilon^{-2} \log n/(wp_i)) \) segments. The probability that \( i \) is a duplicate during a segment is at most \( \binom{w}{2} p_i^2 \). Hence,

\[ \mathbb{P} \{ \text{monitoring } i \text{ during } j\text{-th segment} \} = \binom{w}{2} p_i^2 \cdot O(\epsilon^{-2} \log n/(wp_i)) \]

\[ = O(n^{2-2/k}p_i \epsilon^{-2} \log n/t) \]

Hence, the total expected number of items being monitored is \( O(n^{2-2/k}\epsilon^{-2} \log n/t) \) as claimed.
3 Connectivity

In this section, we consider the problem of determining whether a graph \( G = (V, E) \) is connected given a sequence of \( t \) random samples (with replacement) from \( E \). For notational convenience, denote the number of nodes by \( n = |V| \) and the number of edges by \( m = |E| \).

Our algorithm uses the sampled edges to simulate classical random walks on the graph. In Section 3.1, we discuss how to simulate (and tightly analyze) random walks in our model. The main technical difficulty is ensuring independence when simulating multiple random walks in parallel. Then, in Section 3.2, we adapt a connectivity algorithm of Feige [19] to achieve the required space/sample trade-off.

3.1 Technique: Emulating Classical Random Walks

Consider the following basic algorithm: given a node \( v \), we sample edges until we receive an edge \( \{v, u\} \) for some \( u \). At this point, we move to node \( u \), and repeat. We refer to this method as a sampling walk. Note that the expected time to leave \( v \) is \( m/d(v) \) samples \(^2\) where \( d(v) \) is the degree of a node \( v \), and so a single step of a classical random walk may require \( \Omega(m) \) samples if \( v \) has low degree.

An Inefficient Connectivity Algorithm. This basic algorithm already leads to a \( O(\log n) \) space algorithm which uses \( O(m^2n^2) \) samples in expectation. This follows by starting a sampling walk at node 1 and emulating a classical walk until it traverses nodes 2, 3, \ldots, \( n \) in order. The expected length of this walk is \( O(mn^2) \) because the cover time of \( G \), i.e., the expected length of walk until it visits all nodes (see e.g., [26]), is \( O(mn) \) and there are \( n - 1 \) segments in the traversal. Hence, emulating the random walk takes \( O(m^2n^2) \) samples in expectation. The space use is \( O(\log n) \) bits because the algorithm just needs to remember the current node and the furthest node that has been reached in the sequence. In what follows, we will improve upon the number of samples required and generalize to algorithms that use more space.

The Loopy Graph and an Improved Analysis. The first improvement comes via a better analysis. At a node \( v \) with \( d(v) \) neighbors, there are \( d(v) \) possible samples which would result in a move, and \( m - d(v) \) samples which would not. We can thus view our sampling-based walk on \( G \) as a classical random walk on a new graph \( H \) formed adding \( m - d(v) \) self-loops to each vertex \( v \) in \( G \). We call \( H \) the “loopy graph”.

This view of the sampling walk illuminates its properties. Specifically, \( H \)’s cover time is \( O(mn^2) \) since there are \( mn \) edges and \( n \) nodes. Hence, the above “inefficient” algorithm actually only requires \( (n - 1) \times \text{cover-time}(H) = O(mn^3) \) samples. We will also subsequently use the fact that since \( H \) is \( m \)-regular, its stable distribution is uniform across all nodes.

3.1.1 Multiple Independent Random Walks.

Random walks experience dramatic speedups in cover time, hitting time, etc., when they are split into multiple shorter walks; [27] provides a recent survey and results. These speedups naturally

\(^2\)This is because the number of samples is geometrically distributed with parameter \( d(v)/m \).
require the walks to be independent. In this section, we consider performing \( p \leq n \) random walks in parallel, with the starting point of each walk chosen independently and uniformly from the nodes (and thus according to the stationary distribution on \( H \)). Running these \( p \) walks will require \( O(p \log n) \) space.

The main theorem of this section establishes that it is possible to efficiently perform \( p \) independent, parallel walks in \( H \).

**Theorem 3.1.** Given \( p \leq n \) parallel random walks in \( H \), each starting at an independently-chosen uniformly random node, we can simulate one independent step of each walk using \( O(\log n / \log \log n) \) total samples.

**Issue 1: Multiple Walks can use a Sampled Edge.** The first issue we encounter is that a single sample may be a valid move for multiple walks. If we allow multiple walks to use the same sample, we introduce obvious dependence; if we only allow one of our walks to use the sample, we are “slowing down” walks that have collisions, and again introducing dependence.

When multiple walks are at the same node, we will handle them independently in the following way. We partition the \( p \) walks into \( B_1 \cup B_2 \cup \ldots \cup B_r \) where each \( B_i \) contains at most one walk at each node. We process each batch in turn and hence the total number of samples required equals the number of samples required for a batch multiplied by the number of batches. The next lemma establishes that it suffices to consider \( r = O(\log n / \log \log n) \) batches.

**Lemma 3.2.** With high probability, no node ever contains more than \( O(\log n / \log \log n) \) walks.

**Proof.** Consider a fixed node at a fixed time. Let \( Z \) be the number of walks in this node. Note that \( Z \sim Bin(p, 1/n) \) since each walk is independent and is equally likely to be at any node. Hence \( \mathbb{E}[Z] = p/n \leq 1 \). By an application of the Chernoff bound,

\[
P[Z \geq c \log n / \log \log n] \leq n^{-10}
\]

for some large constant \( c \). The lemma follows by taking the union bound over the \( n \) nodes and \( \text{poly} n \) time-steps.

Henceforth, we assume that at most one walk is at each node, i.e., we analyze how many samples are required to process a single batch. The remaining case where a sampled edge may be valid for multiple walks is if there are walks at both endpoints. To solve this problem, we randomly orient each sampled edge so that it is valid for only one walk. This increases the expected number of samples required by a factor of 2.

**Issue 2: Negative Correlation.** We have reduced the problem to the following situation: we have \( p \) distinct nodes \( u_1, \ldots, u_p \) and can sample arcs \( uv \) uniformly from the set \( E^+ = \{ uv : \{u, v\} \in E_G \} \), i.e., the set of arcs formed by bidirecting each in \( E_G \). Note that \( |E^+| = 2|E_G| \). The goal is to generate a set of arcs \( \{u_1v_1, \ldots, u_pv_p\} \) such that arc is chosen independently and for each \( i \),

\[
v_i = \begin{cases} v \in R : \Gamma(u_i) & \text{with probability } d_G(v_i)/|E^+| \\ u_i & \text{with probability } 1 - d_G(v_i)/|E^+| \end{cases}
\]

where \( \Gamma(u_i) = \{ v : \{u, v\} \in E \} \) is the neighborhood of \( u_i \) in \( G \).
Consider the following procedure: draw a single sample \( uv \in_R E^+ \) and, for each \( i \), set \( v_i = v \) if \( u = u_i \), or \( v_i = u_i \) otherwise. This procedure picks each \( v_i \) according to the desired distribution:

\[
\mathbb{P}[v_i = u_i] = 1 - d_G(v_i)/|E^+|
\]

and conditioned on \( \{v_i \neq u_i\} \), \( v_i \) is uniformly chosen from \( \Gamma(u_i) \). Unfortunately, the procedure obviously does not satisfy the independence requirement because the events \( \{u_i = v_i\} \) and \( \{u_j = v_j\} \) are negatively correlated. However, the following theorem establishes that, with only \( O(1) \) samples from \( E^+ \) in expectation, it is possible to sample independently according to the desired distribution.

**Theorem 3.3 (Efficient Parallel Sampling).** There exists an algorithm that returns samples \( (v_1, \ldots, v_p) \) drawn from the desired distribution \([1]\) while using at most \( 2e - 1 \) samples from \( E^+ \) in expectation.

**Proof.** Our algorithm operates in rounds and each round uses at most 2 samples from \( E^+ \). At the beginning of a round, suppose we have already assigned values to \( v_1, \ldots, v_i \) for some \( i \geq 0 \). Then the round proceeds as follows:

1. Sample \( uv \in E^+ \):
   
   (a) If \( u \notin \{u_{i+1}, \ldots, u_p\} \) then set \( v_{i+1} = u_{i+1}, \ldots, v_p = u_p \)
   
   (b) If \( u = u_j \) for some \( j \in \{i + 1, \ldots, p\} \) then sample an additional arc \( wx \in E^+ \)
      
      i. If \( w \in \{u_{i+1}, \ldots, u_{j-1}\} \) then set \( v_{i+1} = u_{i+1}, \ldots, v_{j-1} = u_{j-1}, v_j = u_j \)
      
      ii. If \( w \notin \{u_{i+1}, \ldots, u_{j-1}\} \) then set \( v_{i+1} = u_{i+1}, \ldots, v_{j-1} = u_{j-1}, v_j = v \)

   and we repeat the process until all \( v_1, \ldots, v_p \) have been assigned.

   To analyze the algorithm we define \( T_j = \{u_jv : \{u_j, v\} \in E_G\} \) to be the set of \( d_G(u_j) \) arcs leaving \( u_j \) and note that because \( u_1, \ldots, u_p \) are distinct, \( T_1, \ldots, T_p \) are disjoint. Also define \( A_j \) to be the event that \( \{v_j \neq u_j\} \). Then, in any round in which \( v_j \) hasn’t yet been assigned:

\[
\mathbb{P}[v_j \text{ is assigned and } A_j|v_j \text{ is assigned}] = \frac{|T_j|}{|E^+| - \sum_{k=i+1}^{j-1}|T_k|} = \frac{|T_j|}{|E^+|},
\]

and hence \( v_j \) is chosen according the desired distribution.

We next show that each \( v_j \) is chosen independently. First observe that, conditioned on \( A_j, v_j \) is independent of \( (v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_p) \). Hence, it suffices to show that all \( A_j \) are independent. Note that the RHS of \([2]\) does not depend on decisions made in previous rounds. Hence, we may deduce that \( A_j \) is independent of the outcome of rounds before \( v_j \) is assigned. Hence, for any \( 1 \leq i_1 < i_2 < \ldots < i_r \leq p \),

\[
\mathbb{P}[A_{i_1} \cap \ldots \cap A_{i_r}] = \mathbb{P}[A_{i_1}] \mathbb{P}[A_{i_2}|A_{i_1}] \ldots \mathbb{P}[A_{i_r}|A_{i_1} \cap \ldots \cap A_{i_{r-1}}] = \mathbb{P}[A_{i_1}] \mathbb{P}[A_{i_2}] \ldots \mathbb{P}[A_{i_r}].
\]

The worst case for the expected number of samples is achieved when \( p = |E^+| \) and each set is of size 1. For the algorithm not to terminate in a given round, we need \( u \in \{u_{i+1}, \ldots, u_p\} \) and hence the index of the sampled \( u \) needs to strictly increase over previous rounds. The probability of this happening for \( r \) rounds is \( \binom{m}{r}/m^r \) and the expected number of rounds which don’t terminate is \( \sum_{r=1}^{m} \binom{m}{r}/m^r \leq e - 1 \). Because each non-terminating round involves two samples, the expected total number of samples is thus at most \( 2e - 1 \).
3.2 Connectivity Algorithm and Analysis

Our algorithm adapts a technique of Feige [18] for determining graph connectivity via a two step process. We first test whether $G$ contains any connected components containing $k$ or fewer nodes (for some $k < n$ to be chosen). If all of the connected components of $G$ contain at least $k$ nodes, we choose $O(n \log n/k)$ nodes at random, and verify that they are all connected to each other. Note that we can expect to have chosen a vertex from each connected component. If we find that all of our chosen vertices are connected, we conclude that $G$ is connected; otherwise, we conclude that $G$ is disconnected.

Our connectivity algorithm thus relies on algorithms for two problems: 1) determining whether the graph has any connected components below a certain size, and 2) determining whether a set of nodes is mutually connected in the graph. In the next two sections, we develop algorithms with sample/space tradeoffs for each of these two problems. We then use them in an algorithm for determining whether the graph $G$ is connected.

3.2.1 Finding small components

Our first subproblem is to determine whether the graph has any connected components below a certain size. Given a node $v$, let the set of nodes in the connected component containing $v$ be denoted $cc(v)$.

**Lemma 3.4.** Given a node $v$ of $G$ and a parameter $r$, we can distinguish between the case where $|cc(v)| < r$ and the case where $|cc(v)| > 2r$ with constant probability using $O(mr^2)$ samples and $\tilde{O}(1)$ space.

**Proof.** We perform a sampling walk of length $O(mr^2)$ samples. While performing this walk, we maintain a 1.1-approximation of the number of distinct vertices visited using an $F_0$ estimator [28]. If the estimated number of vertices visited is at least $3r/2$, we conclude that $|cc(v)| \geq r$; otherwise, we conclude that $|cc(v)| \leq 2r$.

If $|cc(v)| \leq r$, we will clearly visit at most $r$ nodes. Our algorithm correctly concludes this so long as the $F_0$ estimator returns the promised approximation. If $|cc(v)| \geq 2r$, we need to argue that in $O(mr^2)$ samples we will hit at least $2r$ distinct nodes (except with constant probability). This follows from a result by Barnes [29 Thm 1.3] that states that for any connected (multi-)graph, it takes $O(MN)$ time in expectation to hit either $N$ distinct nodes or $M$ distinct edges. Using $M = 2mr$ and $N = 2r$ establishes the result.

**Theorem 3.5.** We can determine whether $G$ has a connected component of size less than $2^k$ using $O(p)$ space and $\tilde{O}(2^k \cdot mn/p)$ samples for any $p \leq n$.

**Proof.** Our algorithm has $k$ rounds, each corresponding to a value $r = 1, 2, 4, \ldots, 2^{k-1}$. In each round we reach one of the following two conclusions: 1) $G$ has no connected components with size in the range $[r, 2r]$ or 2) there exists a connected component of $G$ of size $< 3r$. All graphs satisfy at least one of these conclusions. We then determine $G$ has no connected component of size less than $2^k$ if we never reach the first conclusion.

At a given value of $r$, we choose $O(n \log n/r)$ nodes, so that we hit any connected component of at least $r$ nodes with high probability. From each node, we perform $\tilde{O}(1)$ random walks of length...
\(O(mr^2)\) samples; from Lemma 3.4 this will suffice to determine with high probability whether any of these nodes is in a connected component of size \(\leq 2r\).

We choose \(p\) nodes at a time, and perform \(p\) walks in parallel. From Theorem 3.1 we can perform each set of \(p\) walks using \(O(mr^2)\) samples. The number of samples required for each \(r\) value is then \(O\left(\frac{n}{r}mr^2\right) = O(mn r/p)\), and we thus require a total number of samples \(O(mn^2/p)\).

### 3.2.2 Checking mutual connectivity

The remaining subproblem is to determine whether a set of randomly-chosen nodes is mutually connected.

**Lemma 3.6.** We can determine whether a set of \(O(p)\) randomly-chosen nodes is mutually connected in \(G\) using \(O(p)\) space and \(O(mn^2/p^2)\) samples for any \(p \leq n\).

**Proof.** Feige [19] provides a method for testing whether two nodes \(s\) and \(t\) are connected using space \(O(p)\) and using a total of \(O(mn/p)\) random-walk steps. Their algorithm proceeds by choosing \(p\) “landmark” nodes; we then run \(O(\log n)\) random walks from each landmark and from \(s\) and \(t\). Each random walk is of length \(O(mn/p^2)\). During these random walks we build up a union-find data structure indicating which sets of landmark nodes are connected. If at the end of the algorithm, \(s\) and \(t\) are in the same union-find component, we conclude that \(s\) and \(t\) are connected.

Since \(H\) is regular, the landmark selection process chooses each node with equal probability. Using Feige’s algorithm on the \(p\) randomly-chosen landmarks determines whether this set of \(p\) nodes is mutually connected. The graph \(H\) has \(n\) nodes and \(mn\) edges, so from [19] each walk should be of length \(O(mn/p^2)\). Using Theorem 3.1 we can simulate the \(p\) walks with total of \(O(mn^2/p^2)\) samples. \(\Box\)

We are now ready to prove our main connectivity result.

**Theorem 3.7.** Given sampling access to a graph \(G\), we can determine with high probability whether \(G\) is connected using \(O(p \log n)\) space and \(O(mn^2/p^2)\) samples, for any \(p \leq n\).

**Proof.** We use Theorem 3.5 with \(2^k = n/p\) to verify that \(G\) has no connected components of size less than \(n/p\). If it has such a component, then \(G\) is disconnected. If not, we choose \(O(p \log n)\) random vertices, hitting each remaining component with high probability. Using Lemma 3.6 we test that these vertices are mutually connected. Since we have chosen enough vertices to hit every connected component, this suffices to show that the graph is connected. Each of the two subproblems requires \(O(mn^2/p^2)\) samples and \(O(p)\) space, so these are the sample and space requirements of our algorithm. \(\Box\)

### 4 Lower Bound

Our lower bound result relies on a result by Andoni et al. [30] that implies that \(\Omega(t/r^{2.5})\) space is required\(^3\) to distinguish between the following two cases:

\(^3\)We note that an improvement to the work of Andoni et al. [30] was claimed in [1]; however, the proof given in [1] is incorrect and currently not known to be fixable.
Case 1: We observe a sequence of $t$ samples from a distribution $p^{no}$ that is uniform on some subset $S \subseteq [t]$ of size $\Theta(t)$.

Case 2: We observe a sequence of $t$ samples from a distribution $p^{yes}$ such that $p^{yes}_i = r/t$ for some $i \in [t]$ and uniform on some subset $T \subseteq [t] \setminus \{i\}$.

By combining this result with a hashing technique we establish the following result.

**Theorem 4.1.** Any constant factor approximation of $F_k(D)$ given a sequence of $t$ IID samples on $[n]$ requires $\Omega(n^{2.5(1-1/k)} / (\log^{2.5} n \cdot t^{1.5}))$ space.

**Proof.** Let $h : [t] \rightarrow [n]$ be a fully-random hash function and consider the problem of distinguishing $p^{no}$ and $p^{yes}$ where we set $r = c \log n \cdot t \cdot n^{1/k-1}$ for some constant $c > 0$. By applying $h$ on each distribution (i.e., applying $h$ to each observed sample) we generate two new distributions $q^{no}$ and $q^{yes}$ over $[n]$ where:

$$q^{no}_i = \sum_{j : h(j) = i} p^{no}_j$$

and

$$q^{yes}_i = \sum_{j : h(j) = i} p^{yes}_j$$

Note that with high probability $\max_i q^{no}_i = O(\log n \cdot 1/n)$ and hence

$$F_k(q^{no}) \leq n \cdot O((\log n \cdot 1/n)^k) = O(\log^k n \cdot n^{1-k})$$.

However, $\max_i q^{yes}_i \geq r/n$ and so

$$F_k(q^{yes}) \geq r^k / t^k = c^k \cdot \log^k n \cdot n^{1-k}$$.

Hence, for a sufficiently large value of the constant $c > 0$ we can ensure that any constant approximation of $F_k$ distinguishes between $q^{yes}$ and $q^{no}$ and hence, also distinguishes between $p^{yes}$ and $p^{no}$. However, by the result of Andoni et al. [30] we know that this requires $\Omega(t/r^{2.5}) = \Omega(n^{2.5(1-1/k)} / (\log^{2.5} n \cdot t^{1.5}))$ space. 

**References**


A Omitted Proofs

**Lemma A.1.** Let $X \sim NB(r, p)$ be distributed according to the negative binomial distribution with parameters $r \in \mathbb{N}$ and $p \in (0, 1)$. Then,

$$
P[|r/X - p| \geq \epsilon p] \leq 2 \exp \left( \frac{-\epsilon^2 r}{3(1 + \epsilon)} \right)
$$

Hence, for some $c > 0$ and $r = ce^{-2} \log m$, $P[|r/X - p| \geq \epsilon p] \leq m^{-100}$.

**Proof.** Consider $Y \sim Bin(\frac{r}{p(1+\epsilon)}, p)$. Then

$$
P \left[ X < \frac{r}{p(1+\epsilon)} \right] = P[ Y \geq r ] = P[ Y \geq (1 + \epsilon) \mathbb{E}[Y] ] \leq \exp \left( \frac{-\epsilon^2 r}{3(1 + \epsilon)} \right)
$$

Similarly, let $Y \sim Bin(\frac{r}{p(1-\epsilon)}, p)$. Then

$$
P \left[ X > \frac{r}{p(1-\epsilon)} \right] = P[ Y < r ] = P[ Y < (1 - \epsilon) \mathbb{E}[Y] ] \leq \exp \left( \frac{-\epsilon^2 r}{3(1 - \epsilon)} \right)
$$

The result follows from the union bound. \qed