Representation Obliviousness and Pseudodeterminism in Streaming Algorithms

MICHAEL CHEN, Iowa State University, USA
SOURAV CHAKRABORTY, Indian Statistical Institute, India
A. PAVAN, Iowa State University, USA
N. V. VINODCHANDRAN, University of Nebraska–Lincoln, USA

In this work, we study the notion of *representation obliviousness* in the context of pseudodeterministic streaming algorithms. A (randomized) streaming algorithm $\mathcal{A}$ is pseudodeterministic, if for every stream $\mathcal{D}$, there is a "canonical value" $g(\mathcal{D})$ so that with probability at least 2/3, $\mathcal{A}$ on input stream $\mathcal{D}$ outputs $g(\mathcal{D})$. Intuitively, a randomized algorithm is representation oblivious if the output distribution of the algorithm does not depend on the representation of the input. We investigate this notion in the context of streaming algorithms, more specifically, distinct elements estimation (F_0 estimation) in data streams. In this context, representation obliviousness captures the idea that the output distribution of an algorithm for estimating F_0 should only depend on the set of distinct elements of the stream. This is a natural notion, as we note that standard streaming algorithms are representation-oblivious in this sense. We prove that any representation oblivious pseudodeterministic streaming algorithm for estimating F_0 must use $\Omega(n)$ space, where $[n]$ is the universe. More generally, we prove that any representation oblivious pseudodeterministic $t(n)$-pass streaming algorithm requires $\Omega(n/t(n))$ space. This lower bound matches the space requirement of the straightforward multi-pass deterministic algorithm that exactly computes F_0.

CCS Concepts: • **Mathematics of computing → Probabilistic algorithms**; • **Theory of computation → Communication complexity**; **Problems, reductions and completeness**; **Streaming models**.

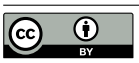
Additional Key Words and Phrases: Streaming, Pseudodeterministic, Distinct Elements Estimation

ACM Reference Format:
Michael Chen, Sourav Chakraborty, A. Pavan, and N. V. Vinodchandran. 2025. Representation Obliviousness and Pseudodeterminism in Streaming Algorithms. *Proc. ACM Manag. Data* 3, 5 (PODS), Article 282 (November 2025), 17 pages. https://doi.org/10.1145/3767718

1 Introduction

In this paper, we introduce and investigate the notion of *representation oblivious* streaming algorithms in the context of *pseudodeterminism*. Intuitively, an algorithm is said to be representation oblivious if the output does not depend on the "representation" of the input. In other words, the algorithm must give the same output (or have the same output distribution in the case of randomized algorithms) for all representations of an input object. The precise definition of representation depends on the problem. To illustrate, consider the problem of estimating distinct elements in a data stream, also known as the F_0 estimation problem. This problem has received considerable attention over the past three decades, leading to the design of (randomized) algorithms with optimal space and update time [1, 3, 6, 14, 18, 27]. It seems intuitive and, indeed, practical to design algorithms

Authors' Contact Information: Michael Chen, Iowa State University, Ames, IA, USA, mqychen@iastate.edu; Sourav Chakraborty, Indian Statistical Institute, Kolkata, WB, India, sourav@isical.ac.in; A. Pavan, Iowa State University, Ames, IA, USA, pavan@iastate.edu; N. V. Vinodchandran, University of Nebraska–Lincoln, Lincoln, NE, USA, vinod@unl.edu.



ACM 2836-6573/2025/11-ART282
https://doi.org/10.1145/3767718

where the output distribution remains consistent across streams that share the same set of distinct elements. Thus, a suitable definition in this scenario is that a randomized algorithm $\mathcal{A}$ for estimating distinct elements is representation oblivious if it produces the same output distribution for any two streams with an identical set of distinct elements. For example, for streams $(1, 2, 1, 1, 3)$ and $(2, 1, 3, 3, 3)$, $\mathcal{A}$ should have the same output distribution since the set of distinct elements for both the streams is $\{1, 2, 3\}$.

Similarly, for the problem of estimating F_k, the k^{th} frequency moment (for $k > 0$), it is reasonable to expect that the algorithm's output depends solely on the frequency vector of the stream. Many popular streaming algorithms for frequency estimation inherently adhere to this principle of representation obliviousness, or they can be adapted to do so. We demonstrate this with several well-known streaming algorithms, including the Gibbons-Tirthapura [18] or the BJKST algorithm for F_0 [3], the AMS sketch for F_2, the AMS algorithm for general F_k [1], and a version of the recent CVM algorithm [10]. While the concept of representation obliviousness merits examination in its own right, our investigation is primarily driven by the aim to understand the capabilities of *pseudodeterministic* streaming algorithms for the problem of estimating frequency moments.

Pseudodeterministic Streaming Algorithms. The notion of *pseudodeterminism* was introduced by Gat and Goldwasser to capture the notion of reproducibility in randomized algorithms [16]. Probabilistic algorithms lack *reproducibility* as two different runs of a probabilistic algorithm can produce two different outputs. An illustrative example is that of generating an n-bit prime number. A straightforward probabilistic algorithm for this problem randomly picks an n-bit positive integer and outputs it if it is a prime number. Due to the density of prime numbers, this simple algorithm succeeds with a non-trivial probability, which can be boosted by repetition. However, two different runs of this algorithm will produce two different prime numbers with high probability. Can we design a probabilistic algorithm that consistently outputs the *same prime number* with high probability on different runs of the algorithm? This question led Gat and Goldwasser to introduce the notion of pseudodeterministic algorithms [16] (originally termed Bellagio algorithms in their paper). A randomized algorithm $\mathcal{A}$ is pseudodeterministic if for every x, there exists a *canonical value* $g(x)$ such that $\Pr[A(x) = g(x)]$ is high. Thus, even though pseudodeterministic algorithms are randomized, they behave similarly to deterministic algorithms: on a given input, they output the same canonical value consistently, similar to deterministic algorithms, with high probability. Since its introduction, the notion of pseudodeterminism and its variations have received considerable attention that led to an exploration of this notion in a variety of settings: search algorithms, approximation algorithm, learning algorithms, parallel algorithms, interactive proofs, sublinear algorithms, and streaming algorithms [2, 4, 5, 7, 11–13, 19–25, 29, 33].

In this paper, we focus on *pseudodeterministic streaming algorithms*. As pseudodeterminism lies between determinism and general randomization, a theoretically and practically significant question is:

Can we develop pseudodeterministic algorithms for critical streaming problems that achieve space complexity comparable to that of the best randomized algorithm for these problems?

The most relevant prior work in this direction includes [4, 23, 24]. Goldwasser, Grossman, Mohanty, and Woodruff [23] initiated the study of pseudodeterministic streaming algorithms. They exhibited gaps between space-complexities of randomized, pseuodeterministic, and deterministic streaming algorithms. They proved, for example, that there is no low-memory pseudodeterministic algorithm for finding a nonzero entry in a vector, while there exists a low-memory randomized algorithm for this problem. They also showed that there exist problems that admit low-memory pseudodeterministic algorithms but no low-memory deterministic algorithms. For frequency-moment estimation, they showed that any sketch-based pseudodeterministic algorithm in the turnstile model

that estimates the stream's ℓ_2 norm requires $\Omega(n)$ space. The classic Morris counter [30] yields an $O(\log \log n)$-space randomized algorithm for estimating stream length; in contrast, [4, 24] show that any pseudodeterministic streaming algorithm for estimating the number of items requires $\Omega(\log n)$ space, matching the deterministic space complexity up to constants. Finally, [9] establishes space lower bounds for pseudodeterministic streaming algorithms for the missing-elements problem, matching deterministic space complexity up to polylogarithmic factors.

Our Contribution. We introduce and formalize the notion of representation oblivious algorithms. For an equivalence relation R on the input space, a randomized algorithm $\mathcal{A}$ is R-oblivious if for all inputs x_1, x_2 such that $(x_1, x_2) \in R$ it holds that the distributions of the output of $\mathcal{A}$ on inputs x_1 and x_2 are the same. When R captures a natural equivalence for the problem, we just say A is representation-oblivious.

We show that any representation oblivious pseudodeterministic streaming algorithm for estimating F_0 requires $\Omega(n)$ space where n is the size of the universe. This asymptotically matches the deterministic upper bound for the exact computation of F_0. In fact, our techniques can be extended to show a tight lower bound for any multi-pass streaming algorithm for F_0: any $t(n)$-pass, representation oblivious pseudodeterministic algorithm for F_0 requires $\Omega(n/t(n))$ space. This matches the deterministic complexity of $t(n)$-pass algorithm for exact computation of F_0. This is the first lower bound on pseudodeterminism in the multi-pass models. Indeed, the authors of [23] raised the question of establishing lower bounds in the multi-pass model as their lower bound techniques can not be extended to multi-pass models. Our results hold for the insertion-only streaming model. We use communication complexity techniques to prove our lower bound.

We argue that standard algorithms for frequency moment estimation are representation oblivious and illustrate that for some well-known algorithms, including the BJKST algorithm for F_0 and AMS sketches for F_2 and F_k. While our focus is to investigate representation obliviousness in the context of pseudodeterminism, we believe the notion of representation obliviousness is worthy of investigation on its own.

Organization. The rest of the paper is organized as follows. In Section 2, we give the necessary notation, definitions, and results from communication complexity that are required to prove our lower bounds. In Sections 2.2 and 2.3, we introduce the notion of representation oblivious algorithms in the context of streaming, pseudodeterminism and communication protocols. In Section 3, we establish the single pass space F_0 lower bound and then we extend it to multi-pass space F_0 lower bound in Section 4, we also show that this lower bound is tight. In Section 5, we illustrate the representation obliviousness of some of the well-known algorithms from the literature. Finally, in Section 6, we give concluding remarks. We also have a discussion on related open problems in the conclusion section.

Related Works. A similar notion called *Path Independence* (PI) for deterministic turnstile streaming algorithms was defined in [15]. We compare PI to our notion of Representation Oblivious (RO) algorithms. A deterministic turnstile streaming algorithm is said to be Path Independent if the algorithm's resultant configuration depends only on the frequency vector of the input and the current configuration. Later in [28], path independence was generalized to randomized turnstile streaming algorithms. A randomized turnstile streaming algorithm is said to be Path Independent if for all possible choices of randomness R, the deterministic algorithm obtained by fixing the random bits to R is path independent. While the notions of Path Independence and Representation Obliviousness are related, they are incomparable. For example, in the context of F_0, RO requires inputs with the same set of distinct elements to have identical output distributions. Whereas PI

requires that for each choice of randomness, the resulting deterministic algorithm's output depends on the frequency vector. Thus, it is possible for an algorithm to be RO but not PI.

In [15, 28], they show how to convert an arbitrary turnstile streaming algorithm into path independent one. However, their transformations crucially require that the streaming algorithm be a turnstile streaming algorithm, and the conversion fails when the algorithm is for the insertion-only model.

2 Preliminaries

We use $\mathbb{N} = \{0, 1, 2, \ldots\}$ to denote the set of natural numbers. For a set X we use 2^X to represent the power set of X and $|X|$ to denote the cardinality of the set. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots n\}$. We use $\psi : X \rightharpoonup Y$ to denote that ψ is a (potentially partial) function from X to Y. If ψ is partial, we use $\mathrm{dom}\,\psi$ to denote the domain of ψ.

We consider randomized algorithms, and often view them as probability measures on the output space, i.e., given a randomized algorithm $\mathcal{A}$ and an input x, $\mathcal{A}(x)(\cdot)$ is a probability measure. Thus, two inputs x, y have the same output distribution, denoted $\mathcal{A}(x) = \mathcal{A}(y)$, if for every output w, $\mathcal{A}(x)(w) = \mathcal{A}(y)(w)$. We also view $\mathcal{A}(x)$ as a random variable wherever convenient.

2.1 Data Streaming Model

We use $[n]$ to denote the universe. A stream over $[n]$ is an element of $[n]^*$, a sequence of items from $[n]$. In the data streaming model, an algorithm $\mathcal{A}$ for a computational task is given one-way access (one pass) to the stream $\mathcal{D}$. The design goal is to estimate a desired function of the stream using as little memory and update time as possible, typically polylogarithmic in universe size n and stream length $|\mathcal{D}|$. We also consider multi-pass models. In this model, the algorithm is allowed to make multiple passes over the input stream: a $t(n)$-pass algorithm is allowed to make $t(n)$, one-way passes over the entire stream before outputting the answer. Refer to [8] for more details on data streaming model and algorithms. In this paper, we will focus mainly on the task of estimating frequency moments of the stream.

We assume the streaming algorithm knows the size of the universe (as input). For a streaming algorithm $\mathcal{A}$ and $n \in \mathbb{N}$, we use $\mathcal{A}_n$ to denote the algorithm has been instantiated with universe $[n]$, i.e., the inputs to $\mathcal{A}_n$ are streams from $[n]^*$.

Let $\mathcal{D}$ be a data stream over $[n]$, for $i \in [n]$ we use $f_i(\mathcal{D})$ to denote the frequency of i in $\mathcal{D}$. We use $f(\mathcal{D}) = (f_1(\mathcal{D}), \ldots, f_n(\mathcal{D}))$ to denote the frequency vector of $\mathcal{D}$. For $k \in \mathbb{N}$, we use $f^k(\mathcal{D}) = (f_1^k(\mathcal{D}), \ldots, f_n^k(\mathcal{D}))$ to represent the k^{th} moment frequency vector, where 0^0 is interpreted as 0. We use $\mathsf{F}_k(\mathcal{D})$ to denote the k^{th} frequency moment of $\mathcal{D}$ which is defined as

$$\mathsf{F}_k(\mathcal{D}) = \sum_{i=1}^{n} f_i^k(\mathcal{D})$$

Note that $\mathsf{F}_0(\mathcal{D})$ is the number of distinct elements of $\mathcal{D}$ and $\mathsf{F}_1(\mathcal{D})$ is the number of items of $\mathcal{D}$.

Definition 2.1. Let $\epsilon, \delta > 0$. A streaming algorithm $\mathcal{A}$ is said to (ϵ, δ)-approximate F_k if for every universe size $n \in \mathbb{N}$ and for every stream $\mathcal{D} \in [n]^*$

$$\Pr\left(|\mathcal{A}_n(\mathcal{D}) - \mathsf{F}_k(\mathcal{D})| \leq \epsilon \mathsf{F}_k(\mathcal{D})\right) \geq 1 - \delta$$

For simplicity, in the rest of the paper we fix $\delta = 1/3$. We will also use the following additional notation: For a set $S \subseteq [n]$, $\mathcal{D}(S)$ denotes the stream representation of S in lexicographic order. For any two streams $\mathcal{D}_1$ and $\mathcal{D}_2$, let $\mathcal{D}_1\mathcal{D}_2$ denote the concatenation of the two streams.

2.2 Pseudodeterministic and Representation Oblivious Streaming Algorithms

We are interested in two notions; pseudodeterminism and representation obliviousness, in the context of streaming.

Definition 2.2 (Pseudodeterministic algorithms [16]). A randomized algorithm $\mathcal{A}$ is **pseudodeterministic** (PD for short) if for all inputs x, there is a canonical value $g(x)$ such that $\Pr(\mathcal{A}(x) = g(x)) \geq \frac{2}{3}$.

In the context of streaming algorithms for F_k, a pseudodeterministic algorithm for a given input stream should output a canonical approximation for F_k with high probability.

We next introduce the notion of representation oblivious algorithms. Let $k \in \mathbb{N} \setminus \{0\}$, note that the frequency moment F_k is "oblivious" to the arrangement of the input stream in the sense that for any stream $\mathcal{D}$, $\mathsf{F}_k(\mathcal{D})$ depends only on the frequency vector $f(\mathcal{D})$. In this sense, for example, the streams $(1, 2, 2, 1, 1)$ and $(2, 1, 1, 2, 1)$ are equivalent. We define this formally next.

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be streams over universe $[n]$. We say $\mathcal{D}_1$ and $\mathcal{D}_2$ are R_k-equivalent if $f^k(\mathcal{D}_1) = f^k(\mathcal{D}_2)$. This equivalence relation partitions the set of inputs to equivalence classes which we call R_k-equivalence classes. When $k = 0$, the condition $f^0(\mathcal{D}_1) = f^0(\mathcal{D}_2)$ is equivalent to the condition that the set of unique elements in $\mathcal{D}_1$ is equal to the set of unique elements in $\mathcal{D}_2$. When $k > 0$, the condition $f^k(\mathcal{D}_1) = f^k(\mathcal{D}_2)$ is the same as $f(\mathcal{D}_1) = f(\mathcal{D}_2)$.

Definition 2.3 (R-oblivious). Let R be an equivalence relation on the input space. A randomized algorithm $\mathcal{A}$ is R-oblivious if for all inputs x_1, x_2 such that $(x_1, x_2) \in R$ it holds that the distributions of the output of $\mathcal{A}$ on inputs x_1 and x_2 are the same.

Definition 2.4. [Representation Oblivious Algorithm for F_k] Let $k \in \mathbb{N}$. Two streams $\mathcal{D}_1$ and $\mathcal{D}_2$ are R_k-equivalent if $f^k(\mathcal{D}_1) = f^k(\mathcal{D}_2)$. A streaming algorithm $\mathcal{A}$ for F_k is said to be **representation oblivious** if for every universe size n, $\mathcal{A}_n$ is R_k-oblivious.

Recall that $\mathcal{A}_n$ is the streaming algorithm $\mathcal{A}$ instantiated with the universe as $[n]$. By combining the notions of pseudodeterminism and representation obliviousness, we arrive at the following definition.

Definition 2.5. A streaming algorithm for F_k is **Representation Oblivious Pseudodeterministic (ROPD)** if it is both representation oblivious and pseudodeterministic.

The above definition means the following: Suppose $\mathcal{A}$ is an ROPD algorithm for F_k. Since $\mathcal{A}$ is pseudodeterministic for every $\mathcal{D}$ there is a canonical value $g(\mathcal{D})$ (which is an ϵ-approximation of $\mathsf{F}_0(\mathcal{D})$) such that $\mathcal{A}$ outputs $g(\mathcal{D})$ with probability at least $2/3$. Representation obliviousness imposes that if $\mathcal{D}_1$ and $\mathcal{D}_2$ belong to the same R_k-equivalence class, then $g(\mathcal{D}_1) = g(\mathcal{D}_2)$.

We now state a more relaxed definition of representation obliviousness.

Definition 2.6 (α-Representation Oblivious Algorithm for F_k). Let $k \in \mathbb{N}$ and $\alpha \geq 0$. A streaming algorithm $\mathcal{A}$ for F_k is said to be **α-representation oblivious** if for every universe size n, and all streams $\mathcal{D}_1$ and $\mathcal{D}_2$ that are R_k equivalent, the statistical distance between $\mathcal{A}_n(\mathcal{D}_1)$ and $\mathcal{A}_n(\mathcal{D}_2)$ is at most α.

We use this relaxed definition to prove that NVGMPC algorithm for F_0 is $\frac{1}{3}$-representation oblivious in Section 5.

Definition 2.7. A streaming algorithm for F_k is **α-Representation Oblivious Pseudodeterministic (α-ROPD)** if it is both α-representation oblivious and pseudodeterministic.

Note that 0-ROPD is the same as ROPD. We remark that our lower bound results extend to α-ROPD setting.

2.3 Communication Complexity

For our lower bound proofs, we will need results from communication complexity. We focus on randomized communication complexity.

Let $m \in \mathbb{N}$ and $\psi : \{0,1\}^m \times \{0,1\}^m \rightharpoonup \{0,1\}$ be a partial function, called the communication problem. We often view binary strings of length m can as subsets of $[m]$ in the standard way. In the randomized communication model, Alice receives $A \in \{0,1\}^m$, and Bob receives $B \in \{0,1\}^m$ such that $(A,B) \in \mathrm{dom}(\psi)$. Both are unaware of the other's string but have access to the same shared random bits. Alice and Bob are allowed to send messages through a communication channel (in both ways). The goal is for Bob to output $\psi(A,B)$. If the input is outside the domain, Bob is allowed to output any arbitrary value. Alice and Bob are allowed to agree on a strategy before receiving the input. The cost of a protocol is the sum of the lengths of messages communicated by Alice and Bob over all inputs in $\mathrm{dom}\,\psi$ and over all random bits. The communication complexity of ψ, denoted $\mathbf{CC}(\psi)$, is the minimum cost over all protocols. No computation restrictions are placed for Alice and Bob. Note that $\mathbf{CC}(\psi) \leq m$ because Alice could simply send her entire input A to B, and then Bob can simply compute $\psi(A,B)$. If Alice and Bob weren't allowed to share public randomness, a protocol that uses private randomness can be converted to one with public randomness with an additive $O(\log m)$ factor via Newman's lemma [32].

We now define the notion of pseudodeterminism for partial functions.

Definition 2.8 (Pseudodeterministic Protocols). Let $m \in \mathbb{N}$. Let $\psi : \{0,1\}^m \times \{0,1\}^m \rightharpoonup \{0,1\}$ be a partial function. A randomized protocol $\mathcal{P}$ for ψ is **pseudodeterministic** (PD) if for **for all inputs** $(A,B) \in 2^{[m]} \times 2^{[m]}$ there exists a canonical value $g(A,B)$ such that

- The protocol $\mathcal{P}$ outputs $g(A,B)$ with probability at least $2/3$.
- $g(A,B) = \psi(A,B)$ if $(A,B) \in \mathrm{dom}(\psi)$.

Note that the above definition requires that even when $(A,B) \notin \mathrm{dom}(\psi)$, the protocol outputs a canonical value with high probability. This subtlety may not be apparent at an initial glance. Suppose $\mathcal{P}$ solves $\psi : 2^{[m]} \times 2^{[m]} \rightharpoonup \{0,1\}$, suppose there exists $(A,B) \in 2^{[m]} \times 2^{[m]}$ but $(A,B) \notin \mathrm{dom}(\psi)$ with $\Pr(\mathcal{P}(A,B) = 0) = \frac{1}{2}$ and $\Pr(\mathcal{P}(A,B) = 1) = \frac{1}{2}$. Therefore, $\mathcal{P}$ is not pseudodeterministic, but still solves ψ.

We next define the notion of representation oblivious protocols.

Definition 2.9 (T-oblivious). Let T be an equivalence relation on the input space $\{0,1\}^m \times \{0,1\}^m$. A randomized protocol $\mathcal{P}$ is T-oblivious if for all inputs (A_1,B_1) and (A_2,B_2) that belong to the same equivalence class, the output distribution of $\mathcal{P}$ on inputs (A_1,B_1) and (A_2,B_2) is the same.

Finally, we define the notion of representation oblivious pseudodeteriministic protocols.

Definition 2.10. Let $\psi : \{0,1\}^m \times \{0,1\}^m \rightharpoonup \{0,1\}$ be a partial function and T be an equivalence relation over $\{0,1\}^m \times \{0,1\}^m$. A communication protocol $\mathcal{P}$ for ψ is **T-oblivious pseudodeteriministic (T-OPD)**, if it is T-oblivious and pseudodeterministic.

We now define some communication problems that we use in this paper.

*Definition 2.11 (*DISJ *[26, 34]).* Let $m \in \mathbb{N}$. The communication problem DISJ_m is the total function $\mathrm{DISJ}_m : 2^{[m]} \times 2^{[m]} \to \{0,1\}$ defined as

$$\mathrm{DISJ}_m(A,B) = \begin{cases} 1 & \text{if } A \cap B = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

*Definition 2.12 (*Promise-Union*).* Let $m \in \mathbb{N}$. The communication problem Promise-Union$_m$ is the partial function Promise-Union$_m : 2^{[m]} \times 2^{[m]} \rightharpoonup \{0,1\}$ defined as

$$\textsc{Promise-Union}_m(A,B) = \begin{cases} 1 & \text{if } |A \cup B| \geq \frac{3m}{4} \\ 0 & \text{if } |A \cup B| \leq \frac{m}{4} \end{cases}$$

We are interested in the oblivious and pseudodeterministic protocols for Promise-Union. For this, we define a specific equivalence relation. Note that a single input instance for the Promise-Union$_m$ is (A, B) where $A \subseteq [m]$ and $B \subseteq [m]$.

Definition 2.13. Let $m \in \mathbb{N}$. Two input instances to Promise-Union$_m$ (A_1, B_1) and (A_2, B_2) are ρ_m-equivalent if $A_1 \cup B_1 = A_2 \cup B_2$.

For the rest of the paper, we are interested in the above relationship ρ_m. From now on, an ROPD protocol for Promise-Union$_m$ refers to a ρ_m-OPD protocol.

3 F_0 Lower bound

In this section, we prove an $\Omega(n)$ space lower bound for estimating F_0 over the universe $[n]$ for ROPD one-pass streaming algorithms.

Theorem 3.1. *Let $\epsilon, \delta \leq \frac{1}{3}$. Any ROPD streaming algorithm that (ϵ, δ)-approximates F_0 requires $\Omega(n)$ space (where n is the size of the universe).*

Our proof approach shall be to show that if an ROPD F_0 estimator using $o(n)$-space exists, then there is a $o(m)$ communication protocol to solve Disj$_m$ problem via the reductions depicted below. This is a contradiction because the communication complexity of Disj$_m$ is $\Omega(m)$ [26, 34].

ROPD F_0 estimator

$\Downarrow$

ROPD communication protocol for Promise-Union

$\Downarrow$

Communication protocol for Disj

Lemma 3.2. *For any $n \in \mathbb{N}$, if there is a ROPD streaming algorithm that $(\frac{1}{3}, \frac{1}{3})$-approximates F_0 (with universe size n) using space s then there is a ROPD protocol for the* Promise-Union$_n$ *with communication complexity s.*

Proof. Let $n \in \mathbb{N}$. Let $\mathcal{A}_n$ be a ROPD streaming algorithm that $(\frac{1}{3}, \frac{1}{3})$-approximates F_0 over a universe $[n]$ and uses space s. We now define the protocol $\mathcal{P}_n$ to solve the Promise-Union$_n$:

Alice receives $A \subseteq [n]$ and Bob receives $B \subseteq [n]$. They convert their input sets to streams $\mathcal{D}_A = \mathcal{D}(A)$ and $\mathcal{D}_B = \mathcal{D}(B)$, respectively. Alice runs the streaming algorithm $\mathcal{A}_n$ on $\mathcal{D}_A$ and then communicates the state at the end of her execution to Bob (requiring only s bits of communication). On receiving this state, Bob continues the execution with his part of the stream $\mathcal{D}_B$. Let $\mathcal{D} = \mathcal{D}_A\mathcal{D}_B$. At the end of the simulation, Bob has $\mathcal{A}_n(\mathcal{D})$. Bob outputs 1 if $\mathcal{A}_n(\mathcal{D})$ is at least $n/2$; otherwise, Bob outputs 0.

We now prove that $\mathcal{P}_n$ (1) is pseudodeterministic, (2) is representation oblivious, and (3) solves Promise-Union$_n$

Claim 3.3. *$\mathcal{P}_n$ is pseudodeterministic.*

Proof. For a stream $\mathcal{D}$ over $[n]$, let $g(\mathcal{D})$ be the canonical value of $\mathcal{A}_n$, i.e., $\Pr(\mathcal{A}_n(\mathcal{D}) = g(\mathcal{D})) \geq \frac{2}{3}$. For $A, B \subseteq [n]$, define

$$h(A,B) = \begin{cases} 1 & \text{if } g(\mathcal{D}(A)\mathcal{D}(B)) \geq \frac{n}{2} \\ 0 & \text{if } g(\mathcal{D}(A)\mathcal{D}(B)) < \frac{n}{2} \end{cases},$$

so that, $\Pr(\mathcal{P}(A,B) = h(A,B)) \geq \frac{2}{3}$. Thus, h testifies that $\mathcal{P}_n$ is pseudodeterministic. We emphasize that the pseudodeterminism holds for *all* inputs. □

Claim 3.4. *$\mathcal{P}_n$ is representation oblivious.*

Proof. Let $(A_1, B_1), (A_2, B_2)$ be inputs (they need not be part of the promised region) such that (A_1, B_1) and (A_2, B_2) are ρ_n-equivalent, i.e., $A_1 \cup B_1 = A_2 \cup B_2$. We show that the output distributions are the same. Let $\mathcal{D}_1 = \mathcal{D}_{A_1}\mathcal{D}_{B_1}$ and $\mathcal{D}_2 = \mathcal{D}_{A_2}\mathcal{D}_{B_2}$. As the inputs are Promise-Union$_n$ equivalent we have that $f^0(\mathcal{D}_1) = f^0(\mathcal{D}_2)$, because the unique elements in $\mathcal{D}_1$ and $\mathcal{D}_2$ are the same and as a consequence $\mathcal{D}_1$ and $\mathcal{D}_2$ are R_0-equivalent. We have $\mathcal{A}_n(\mathcal{D}_1) = \mathcal{A}_n(\mathcal{D}_2)$ because $\mathcal{A}_n$ is representation oblivious. Which implies that $\mathcal{P}_n(A_1, B_1) = \mathcal{P}_n(A_2, B_2)$, as required. □

Claim 3.5. *$\mathcal{P}_n$ solves* Promise-Union$_n$.

Proof. Let $(A, B) \in \mathrm{dom}(\text{Promise-Union}_n)$ be a promised input, i.e., either $|A \cup B| \leq \frac{n}{4}$ or $|A \cup B| \geq \frac{3n}{4}$. Also, let $\mathcal{D} = \mathcal{D}(A)\mathcal{D}(B)$, the concatenation of the stream of the set A and B. As discussed in Claim 3.3, the output of the protocol $\mathcal{P}_n$ will depend upon the value of $\mathcal{A}_n(\mathcal{D})$. To prove $\mathcal{P}_n$ solves Promise-Union$_n$ it suffices to show that $g(\mathcal{D})$ is less than $\frac{n}{2}$ for the case when $|A \cup B| \leq \frac{n}{2}$ and $g(\mathcal{D})$ is at least $\frac{n}{2}$ when $|A \cup B| \geq \frac{3n}{4}$. This suffices because the event $\mathcal{A}_n(\mathcal{D}) = g(\mathcal{D})$ occurs with probability at least 2/3. Since $\mathcal{A}_n$ is an $(\frac{1}{3}, \frac{1}{3})$-approximation algorithm for F_0 we have:

Case 1. If $|A \cup B| \leq \frac{n}{4}$ then $g(\mathcal{D}) \leq \frac{n}{4}(1 + \frac{1}{3}) = \frac{n}{3}$.

Case 2. If $|A \cup B| \geq \frac{3n}{4}$ then $g(\mathcal{D}) \geq \frac{3n}{4}(1 - \frac{1}{3}) = \frac{n}{2}$.

Hence, the canonical values of $\mathcal{P}_n$ gives the correct answer, i.e., h satisfies

$$h(A,B) = \begin{cases} 1 & \text{if } |A \cup B| \geq \frac{3n}{4} \\ 0 & \text{if } |A \cup B| \leq \frac{n}{4} \end{cases}.$$

Thus $\mathcal{P}_n$ solves Promise-Union$_n$. □

Thus, with Claim 3.3, Claim 3.4, and Claim 3.5, we conclude that $\mathcal{P}_n$ is a protocol for Promise-Union$_n$ that is representation oblivious and pseudodeterministic. □

We first state a fact due to [26, 34] to stating that Disj$_m$ has communication complexity $\Omega(m)$.

Fact 3.6 ([26, 34]). $\mathbf{CC}(\text{Disj}_m) = \Omega(m)$

Our goal now is to show that the existence of a ROPD protocol for Promise-Union implies the existence of a PD communication protocol for Disj over a smaller universe. Suppose we want to solve Disj$_m$, distinguishing whether $A \cap B = \emptyset$ or $A \cap B \neq \emptyset$, equivalently distinguishing between $\overline{A} \cup \overline{B} = [m]$ or $\overline{A} \cup \overline{B} \neq [m]$. We consider the problem Promise-Union$_{4m}$, it distinguishes between the case $|A \cup B| \leq m$ and $|A \cup B| \geq 3m$. We shall show that embedded within a ROPD protocol for Promise-Union$_{4m}$ is a PD protocol for Disj$_m$. All Alice and Bob would need to do is take the complement of their input and then follow the protocol for Promise-Union$_{4m}$.

Lemma 3.7. *The communication complexity of ROPD protocol for* Promise-Union$_m$ *is* $\Omega(m)$.

Proof. Assume we have ROPD protocols $\mathcal{P} = (\mathcal{P}_m)_{m\in\mathbb{N}}$ that solves $(\textsc{Promise-Union}_m)_{m\in\mathbb{N}}$ using $o(m)$ communication. Using this, we shall design protocols $(\mathcal{Q}_m)_{m\in\mathbb{N}}$ that solve $(\textsc{Disj}_m)_{m\in\mathbb{N}}$ using $o(m)$ communication contradicting Fact 3.6.

Let $m \in \mathbb{N}$, we design a protocol $\mathcal{Q}_m$ to solve $\textsc{Disj}_m$. Alice and Bob first agree on the communication problem $\textsc{Promise-Union}_n$ with input size $n = 4m$. They shall use the help of $\mathcal{P}_n$ to solve $\textsc{Disj}_m$. Since $\mathcal{P}_n$ is PD there exists a canonical values for $\mathcal{P}_n$, for $A, B \subseteq [n]$ let $g(A, B)$ be the canonical value of $\mathcal{P}_n$, i.e.,

$$\Pr(\mathcal{P}_n(A, B) = g(A, B)) \geq \frac{2}{3}$$

Furthermore, since $\mathcal{P}_n$ is representation oblivious, we have the additional guarantee that equivalent elements (under ρ_m equivalence relation) have the same output distribution. i.e., for any (A_1, B_1) and (A_2, B_2) if $A_1 \cup B_1 = A_2 \cup B_2$ then $\mathcal{P}_n(A_1, B_1) = \mathcal{P}_n(A_2, B_2)$. In particular, their canonical values are the same, $g(A_1, B_1) = g(A_2, B_2)$. Furthermore, the canonical value only depends on the union of the subsets. For any $X \subseteq [n]$, define $h(X) = g(X, \emptyset)$. We have that h satisfies the following property: for all $A, B \subseteq [n]$, $g(A, B) = h(A \cup B)$. Observe that for any set $X \subseteq [n]$, if $|X| \leq n/4$ then $h(X) = 0$ and if $|X| \geq 3n/4$ then $h(X) = 1$ because $\mathcal{P}_n$ solves $\textsc{Promise-Union}_n$. Therefore, there must exist a minimal Y such that $h(Y) = 1$ and removing any subset from Y results in an output of 0. Furthermore, it must be the case that $|Y| > n/4$. Formally there exists Y with $|Y| > n/4$ such that $h(Y) = 1$ and for every Z such that $Z \subsetneq Y$, $h(Z) = 0$. Fix the lexicographically smallest Y so that $|Y| = n'$, without loss of generality, assume $Y = [n']$, as Alice and Bob can always agree to rearrangements. Note that we have $n' > m$. Alice and Bob agree to use $\mathcal{P}_n$ and Y to solve $\textsc{Disj}_m$.

For $X \subseteq [n']$, we denote $\overline{X} = [n'] \setminus X$, i.e., complement is taken over the universe $[n']$. Identifying disjointness can equivalently be seen as distinguishing between a "full universe" and "not a full universe" by complementing the inputs over the universe $[n']$. Formally, via De Morgan's law, for all $A, B \subseteq [n']$, $A \cap B = \emptyset \iff \overline{A} \cup \overline{B} = [n']$.

We now define the protocol $\mathcal{Q}_m$: Alice receives $A \subseteq [m]$ and Bob receives $B \subseteq [m]$. Note that trivially $A, B \subseteq [n']$ as $n' > m$. Alice creates an instance $A' = \overline{A}$, and Bob creates an instance $B' = \overline{B}$. They proceed as though this were an instance of $\textsc{Promise-Union}_n$ with input (A', B'). After finishing the simulation, Bob obtains $\mathcal{P}_n(A', B')$. If $\mathcal{P}_n(A', B') = 1$ then Bob outputs 1; otherwise, Bob outputs 0.

Correctness follows from the fact that for any set $A, B \subseteq [m]$

$$A \cap B = \emptyset \iff \overline{A} \cup \overline{B} = [n'] \iff h(\overline{A} \cup \overline{B}) = 1$$

The communication required for this protocol is $o(n) = o(4m) = o(m)$. Therefore, if there existed ROPD protocols for $\textsc{Promise-Union}$ with $o(m)$ communication, we could use them to generate protocols for $\textsc{Disj}$ with $o(m)$ communication, which contradicts Fact 3.6. □

We are now ready to conclude Theorem 3.1.

Proof of Theorem 3.1. Let $\epsilon, \delta \leq \frac{1}{3}$. If there exists ROPD streaming algorithm that (ϵ, δ)-approximates F_0 using space $o(n)$ then setting $n = m$ and using Lemma 3.2 there exists ROPD protocols with $o(m)$ communication cost for $\textsc{Promise-Union}$. This directly contradicts Lemma 3.7. □

4 Multipass F_0 Lower Bound

We now extend the result to allow for the streaming algorithm to make multiple passes of the input.

Theorem 4.1. *Let $t : \mathbb{N} \to \mathbb{N}$ and $\epsilon, \delta \leq \frac{1}{3}$. Any ROPD $t(n)$-pass streaming algorithm that (ϵ, δ)-approximates F_0 requires $\Omega(\frac{n}{t(n)})$ space (where n is the size of the universe).*

The proof of theorem 4.1 approach is similar to a single-pass streaming algorithm. We show that a space-efficient ROPD multi-pass streaming algorithm for F_0 can be used to solve Promise-Union efficiently.

Lemma 4.2. *For any $n \in \mathbb{N}$, if there is a ROPD $t(n)$-pass streaming algorithm that $(\frac{1}{3}, \frac{1}{3})$-approximates F_0 (with universe size n) using space s then there is a ROPD protocol for the* Promise-Union$_n$ *with communication complexity $O(s \cdot t(n))$.*

Proof. Let $n \in \mathbb{N}$. Let $\mathcal{A}_n$ be a ROPD $t(n)$-pass streaming algorithm that $(\frac{1}{3}, \frac{1}{3})$-approximates F_0 over a universe $[n]$ and uses space s. We now define the protocol $\mathcal{P}_n$ to solve the Promise-Union$_n$:

Alice receives $A \subseteq [n]$ and Bob receives $B \subseteq [n]$. They convert their input sets to streams $\mathcal{D}_A = \mathcal{D}(A)$ and $\mathcal{D}_B = \mathcal{D}(B)$, respectively. Alice runs the streaming algorithm $\mathcal{A}_n$ on $\mathcal{D}_A$ and then communicates the state at the end of her execution to Bob. On receiving this state, Bob continues the execution with his part of the stream $\mathcal{D}_B$, he then returns the algorithm's state to Alice. Alice then initializes the next pass of the algorithm, and then they repeat the process for a total of $t(n)$ rounds of communication. Let $\mathcal{D} = \mathcal{D}_A\mathcal{D}_B$, after $t(n)$ rounds of communication (and therefore they have simulated $t(n)$ passes of the stream), Bob has $\mathcal{A}_n(\mathcal{D})$. Bob outputs 1 if $\mathcal{A}_n(\mathcal{D})$ is at least $n/2$; otherwise, Bob outputs 0.

The cost of this communication is $2s \cdot t(n) = O(s \cdot t(n))$. Using identical arguments from Claim 3.3, Claim 3.4, and Claim 3.5, we conclude that $\mathcal{P}_n$ is a protocol for Promise-Union$_n$ that is representation oblivious and pseudodeterministic. □

Proof of Theorem 4.1. Let $t : \mathbb{N} \to \mathbb{N}$ and $\epsilon, \delta \leq \frac{1}{3}$. If there exists ROPD $t(n)$-pass streaming algorithm that (ϵ, δ)-approximates F_0 using space $s = o(\frac{n}{t(n)})$ then setting $n = m$ and using Lemma 4.2 there exists ROPD protocols with $o(s \cdot t(n)) = o(\frac{n}{t(n)} \cdot t(n)) = o(n)$ communication cost for Promise-Union. This directly contradicts Lemma 3.7. □

Note that Theorem 4.1 is indeed tight as one can design a $t(n)$-pass representation oblivious deterministic $t(n)$-pass streaming algorithm that computes F_0 of any stream using $O(\frac{n}{t(n)})$ space.

Let $t : \mathbb{N} \to \mathbb{N}$. We now describe a simple representation oblivious deterministic $t(n)$-pass streaming algorithm to that computes F_0. For simplicity's sake, assume $t(n)$ divides n. Partition the $[n]$ into $B_1, B_2, \ldots, B_{t(n)}$ where B_1 first $\frac{n}{t(n)}$ elements of $[n]$, B_2 are the next $\frac{n}{t(n)}$ elements of $[n]$, and so on. We have $|B_p| = \frac{n}{t}$.

Algorithm 1: Representation Oblivious Deterministic $t(n)$-Pass Streaming Algorithm for F_0

Input: Stream $\mathcal{D} = (a_1, \ldots, a_m)$

$D \leftarrow 0$
for $p = 1, \ldots, t(n)$ **do**
  Observe $\mathcal{D}$ and count the size of the set $\{a_i \mid i \in [m]\} \cap B_p$ using $O(\frac{n}{t(n)})$ bits
  $D_p \leftarrow$ Number of distinct elements in $\mathcal{D}$ that belong to B_p
  $D \leftarrow D + D_p$
return D

Observation 4.3. *For any $t : \mathbb{N} \to \mathbb{N}$, Algorithm 1 is a $t(n)$-pass representation oblivious deterministic $t(n)$-pass streaming algorithm that computes exact F_0 using $O(\frac{n}{t(n)})$ space.*

Proof. Clearly, the presented algorithm is deterministic (as a consequence, pseudodeterministic) and uses $O(\frac{n}{t(n)})$ space because in each pass, the space used for counting can be re-used.

We now prove correctness, let $\mathcal{D} = (a_1, \ldots, a_m)$ be a stream and let $A = \{a_i \mid i \in [m]\}$ be the set of unique elements. The set A can be partitioned as $A = \bigcup_{p=1}^{t(n)} (A \cap B_p)$. In each pass $p \in [t(n)]$, D_p is computed deterministically which is exactly $|A \cap B_p|$.

Representation Obliviousness follows from the fact that Algorithm 1 computes exact F_0 deterministically, and from the fact that F_0 of a stream only depends on the number of unique elements. Formally, let $\mathcal{D}_1, \mathcal{D}_2$ be equivalent streams, i.e., $f^0(\mathcal{D}_1) = f^0(\mathcal{D}_2)$. Because of correctness of algorithm, the output is exactly $\mathsf{F}_0(\mathcal{D}_1) = \mathsf{F}_0(\mathcal{D}_2)$ on either input. □

5 Illustration of Representation Oblivious Streaming Algorithms

We argue that known algorithms for F_k are representation oblivious and illustrate it with some popular algorithms. We note that if we have an unbiased estimator that is representation oblivious, the algorithm will remain representation oblivious even after employing standard boosting techniques. For this section, we use m to denote the stream length.

The BJKST Algorithm [3] presented below is a randomized streaming algorithm that given $\epsilon > 0$, approximates F_0 of any stream within a $1 \pm \epsilon$ multiplicative factor with probability at least $\frac{2}{3}$ using $O(\frac{1}{\epsilon^2} \log n)$ space. We shall prove that the BJKST algorithm for F_0 is representation oblivious.

Algorithm 2: BJKST Algorithm for F_0 [3]

Input: Universe size n, $\epsilon > 0$, Stream $S = (a_1, \ldots, a_m)$

1 Randomly pick $h : [n] \to [N = n^3]$ from 2-universal hash family $\mathcal{H}$.
2 $t \leftarrow O(\frac{1}{\epsilon^2})$
3 Observe the stream S and maintain a set V of t smallest hashed values
4 $v \leftarrow t^{th}$ smallest element of V
5 Output $\frac{tN}{v}$

Theorem 5.1. *The* BJKST *algorithm for* F_0 *is representation oblivious.*

Proof. A version of the BJKST algorithm from [3] is presented in Algorithm 2. Let $\mathcal{D}_1 = (a_1, \ldots, a_m), \mathcal{D}_2 = (b_1, \ldots, b_{m'})$ be two streams such that $f^0(\mathcal{D}_1) = f^0(\mathcal{D}_2)$. To prove that the output distributions are the same, it suffices to show that the outputs are the same conditioned on the same hash function being chosen.

Fix a hash function $h \in \mathcal{H}$, let V_1 and V_2 be as described in Algorithm 2 when $\mathcal{D}_1$ and $\mathcal{D}_2$ is given as input respectively. Since $f^0(\mathcal{D}_1) = f^0(\mathcal{D}_2)$ (i.e., the set of unique elements is the same) and the hash function is the same, we have $V_1 = V_2$. Therefore, both outputs have the same value. Hence, the output distributions obtained by uniformly at random picking a hash function in $\mathcal{H}$ are the same. □

The NVGMPC Algorithm [31] (a version of the recent CVM algorithm for F_0 [10]) presented below is a randomized streaming algorithm that given $\epsilon > 0$, approximates F_0 of any stream within a $1 \pm \epsilon$ multiplicative factor with probability at least $\frac{2}{3}$ using $O(\frac{1}{\epsilon^2} \log^2 n)$ space. While the original NVGMPC Algorithm (as presented in [31]) is for a much more general problem, namely the Klee's measure problem or estimating volume of union of Delphic sets, one can apply their algorithm to get an F_0 estimation of a stream of items. We note that the NVGMPC Algorithm (for the F_0 estimation) is not truly representation oblivious - at least not according to the Definition 2.4. However, we shall prove that the NVGMPC Algorithm for the F_0 is $\frac{1}{3}$-representation oblivious.

Algorithm 3: NVGMPC algorithm for F_0 [31]

Input : Universe size n, $\epsilon > 0$, Stream $S = (a_1, \ldots, a_m)$

1 Initialize
2 thresh $\leftarrow \max\{\lceil \log 2/\epsilon \rceil,\ 18\}$; Reps $\leftarrow \lceil 90/\epsilon^2 \rceil$;
3 **for** *(*$1 \le k \le \log n$*) and (*$1 \le r \le$ Reps*)* **do**
4 $\quad \mathcal{X}_r^{(k)} \leftarrow \emptyset$
5 Streaming Phase:
6 **for** *(*$1 \le k \le \log n$*) and (*$1 \le r \le$ Reps*)* **do**
7 $\quad$ For every element of the stream, pick it independently with probability $1/2^k$ and store the first thresh of them in $\mathcal{X}_r^{(k)}$
8 Post-Streaming Phase:
9 **for** $1 \le k \le \log n$ **do**
10 $\quad \overline{Z}^{(k)} = \frac{1}{\text{Reps}} \sum_{r=1}^{\text{Reps}} |\mathcal{X}_r^{(k)}|$
11 $k^* = \max\{k \in [\log n] : \overline{Z}^{(k)} \ge 1\}$
12 **Output** $\overline{Z}^{(k^*)} \cdot 2^{k^*}$

Theorem 5.2. *The* NVGMPC *algorithm for* F_0 *is* $\frac{1}{3}$*-representation oblivious.*

Proof. The algorithm of NVGMPC can be thought of as running $\log n \times$ Reps number of algorithms in parallel - each algorithm sampling from the stream with different probabilities. Finally, after the stream ends, the output of the algorithm is a function of the number of elements picked in the different algorithms.

More specifically, for any $1 \le k \le \log n$ and any $1 \le r \le$ Reps, the algorithm picks each element in the stream with probability $1/2^k$ and stores the first thresh many in the bucket $\mathcal{X}_r^{(k)}$. So at the end of the stream the $|\mathcal{X}_r^{(k)}|$ is, in fact, a random variable which is equal to $\min\{\text{thresh}, \mathcal{Z}_r^{(k)}\}$, where $\mathcal{Z}_r^{(k)}$ is another random variable indicating the number of elements picked if each element is independently picked with probability $1/2^k$. In other words, $\mathcal{Z}_r^{(k)}$ is a random number picked accordingly to the distribution $\mathsf{Bin}(n, 1/2^k)$.

The main contribution of NVGMPC is how they manage to pick each element of the stream independently with probability $1/2^k$. When element a_i comes it is removed from the stored set $\chi_r^{(k)}$ and then a_i is added to $\chi_r^{(k)}$ with probability $1/2^k$. Here we must note that the process is not representation oblivious. The main issue is that if, for any k, the number of elements picked (when each element of the stream is independently with probability $1/2^k$) is more than thresh then only the first thresh of them are stored in $\chi_r^{(k)}$. Thus clearly which elements are stored can be very much be dependent on the ordering in the stream and hence the process (or rather the $|\chi_r^{(k)}|$) is not representation oblivious. The proof of correctness of the algorithm goes via proving that the value of k^* (in line 11) is either k_0 or $k_0 - 1$, where k_0 is the unique number satisfying the following equation

$$\frac{3}{4} < \frac{\mathsf{F}_0(\mathcal{S})}{2^{k_0}} \le \frac{3}{2}.$$

The output of the algorithm depends only on the value of the $|\mathcal{X}_r^{(k)}|$ for $k > k_0 - 1$.

From our point of view, all we need to realize is that with probability at least 2/3 if the value of k^* in (line 11) remains same and the distribution of value of $|\chi_r^{(k^*)}|$ remains same for any ordering of the stream then the the algorithm is almost representation oblivious. And this is true as for $k \geq k_0 - 1$, the number of size of $|\chi_r^{(k)}|$ is always less than thresh and which mean for $k > k_0 - 1$ the the distribution of the random variable $|\mathcal{X}_r^{(k)}|$ is invariant under the permutation of elements in the stream.

Since the final output of NVGMPC is a function of the set of random variables $|\mathcal{X}_r^{(k)}|$ (for $k > k_0 - 1$, the algorithm is indeed almost representation oblivious. □

We note that all sketching algorithms (as defined in [23]) for F_k ($k > 0$) are representation oblivious. A sketching algorithm is a pair $(\mathcal{M}, g)$ where $\mathcal{M}$ is a distribution over $d \times n$ matrices along with a function $g : \mathbb{R}^d \to \mathbb{R}$ where $d \ll n$. Let $\mathcal{D}$ be a data stream and $x = f(\mathcal{D})$ be the frequency vector of $\mathcal{D}$. The output of the sketching algorithm on input $\mathcal{D}$ is $g(Mx)$ where M is drawn according to $\mathcal{M}$. Note that known algorithms for estimating F_2 (such as AMS algorithm from [1]) are sketching algorithms under this definition.

Observation 5.3. *Let $k > 0$. All sketching algorithms [23] for F_k are representation oblivious.*

Proof. Let $k > 0$ and let $(\mathcal{M}, g)$ be a sketching algorithm. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be R_k-equivalent streams, i.e., $f(\mathcal{D}_1) = f(\mathcal{D}_2)$. Clearly, the output distributions are the same as $g(\mathcal{M}f(\mathcal{D}_1)) = g(\mathcal{M}f(\mathcal{D}_2))$. □

The AMS algorithm for F_k [1] below is a randomized streaming algorithm that, given $k > 0$ and $\epsilon > 0$, approximates F_k of any stream within a $1 \pm \epsilon$ multiplicative factor with probability at least $\frac{2}{3}$ using $O(n^{1-\frac{1}{k}})$ space. We note that this is a sampling-based algorithm, rather than sketch-based. We shall prove that the AMS algorithm for estimating F_k is also representation oblivious.

Algorithm 4: AMS Algorithm for F_k [1]

Input: Universe size n, $\epsilon > 0$, Stream $S = (a_1, \ldots, a_m)$

1 **Function** `AMS_thread(`S`)`**:**
2 Pick random $p \sim \text{Unif}([m])$
3 $r \leftarrow \sum_{i=p}^{m} \mathbf{1}_{\{a_i = a_p\}}$ //$\mathbf{1}_{\{\cdot\}}$ is the indicator function
4 **return** $(r^k - (r-1)^k)m$
5 Run $O(\frac{1}{\epsilon^2})$ parallel independent copies of `AMS_thread(`S`)` and output the average.

Theorem 5.4. *The* AMS *algorithm for F_k is representation oblivious.*

Proof. A version of the AMS algorithm for estimating F_k from [1] for the general value of $k > 0$ is presented in Algorithm 4. For the sake of simplicity, we assume that the algorithm also gets the stream's length as input. This assumption can easily be lifted by reservoir sampling, as shown in [1]. It suffices to show that the single instance `AMS_thread` is representation oblivious, as averaging independent copies does not change the representation oblivious nature of the algorithm

Let $\mathcal{D}_1 = (a_1, \ldots, a_m)$, $\mathcal{D}_2 = (b_1, \ldots, b_{m'})$ such that they are R_2-equivalent, i.e., $f(\mathcal{D}_1) = f(\mathcal{D}_2)$. Necessarily, we have $m = m'$, furthermore $\mathcal{D}_1$ is a rearrangement of $\mathcal{D}_2$.

We will show that for every $p \in [m]$, there exists an $\ell \in [m]$ such that the output for $\mathcal{D}_1$ conditioned on p being chosen is the same as the output of $\mathcal{D}_2$ conditioned on ℓ being chosen. Furthermore, this mapping is bijective. Since the index is chosen uniformly at random, the output distribution is therefore the same.

Toward that end, fix a $p \in [m]$, let $x = a_p$. Let $t \in [f_x(\mathcal{D}_1)]$ so that a_p is the t^{th} occurrence of x in $\mathcal{D}_1$. Let $\ell \in [m]$ be the index of the t^{th} occurrence of x in $\mathcal{D}_2$. Observe that this construction leads to the following: $\sum_{i=p}^{m} \mathbf{1}_{\{a_i=a_p\}} = \sum_{i=\ell}^{m} \mathbf{1}_{\{b_i=b_\ell\}}$, i.e., the frequencies of x in $\mathcal{D}_1$ after position p is the same as the frequencies of x in $\mathcal{D}_2$ after position ℓ. Therefore, the output for $\mathcal{D}_1$ when p is chosen is the same as the output for $\mathcal{D}_2$ when ℓ is chosen.

Thus, for every $p \in [m]$, there exists a unique $\ell_p \in [m]$ such that the outputs are the same. Since the index p is chosen uniformly at random, summing over all possible outputs for each p, the output distribution is the same for $\mathcal{D}_1$ and $\mathcal{D}_2$. □

6 Discussion

We defined the notion of representation oblivious algorithms in the context of streaming algorithms and investigated the pseudodeterministic space complexity of such algorithms for estimating F_0. We established that for such algorithms, pseudodeterministic complexity is the same as deterministic complexity: they have a space lower bound of $\Omega(n)$. Therefore, if a $o(n)$-space pseudodeterministic algorithm exists for estimating F_0 of a stream, then it must use the representation of the stream in some non-trivial way. Designing natural algorithms using low space seems to mandate representation obliviousness because known techniques rely on ignoring the structure of the input representation. In particular, current techniques for streaming algorithms are either hashing-based or sampling-based, which are black-box techniques and do not exploit the representation of the stream.

The notion of representation obliviousness is a very natural one. Certainly, the algorithms described in Section 5 could be made non-representation oblivious by (say) perturbing the output depending on the order of the stream: keep track of the smallest and the second smallest and distinguish between the cases when the smallest comes before the second smallest versus the reverse. But that seems like a meaningless computation. A comprehensive investigation of representation obliviousness, even in settings other than streaming, is meaningful.

It is worth noting that if we are allowed the output set to be two values (2-PD), there is a representation oblivious algorithm for F_0 with $O(\log n)$ space (for constant ε and δ): use BJKST algorithm and round the output to the closest multiple of ε (this works for F_k also). Thus, it is very intriguing that while ROPD algorithms for F_0 require $\Omega(n)$ space, there is an $O(\log n)$ space 2-PD algorithm that is representation oblivious.

Discussion on lower bound for estimating F_k. Finally, we would like to discuss some significant open problems that have emerged from our research. An immediate open problem is to establish a space lower bound for representation oblivious pseudodeterministic streaming algorithms for estimating F_k (with $k > 0$), similar to the one we proved for F_0. The crucial component for the lower bound is the communication complexity lower bound for ρ_m-OPD protocol for Promise-Union$_m$. Recall that the equivalence relation ρ_m is defined as: (A_1, B_1) is equivalent to (A_2, B_2) if $A_1 \cup B_1 = A_2 \cup B_2$. To obtain a lower bound for representation oblivious pseudo-deterministic streaming algorithms for estimating F_k (with $k > 0$), it is enough to prove a lower bound for τ_m-OPD protocol for Promise-Union$_m$, where τ_m (a refinement of ρ_m) is defined as follows:

Definition 6.1. Two input instances (A_1, B_1) and (A_2, B_2) are τ_m-equivalent if $A_1 \cup B_1 = A_2 \cup B_2$ and $A_1 \cap B_1 = A_2 \cap B_2$.

Possibly a bit surprisingly, there exists a simple pseudodeterministic communication protocol for Promise-Union$_m$ with respect to the equivalence relation τ_m:

τ_m-OPD protocol for PROMISE-UNION$_m$:
- Alice send size of her input.
- If the size of Alice's input and size of Bob's input add to more than $3m/4$ then Bob outputs 1 else 0

One can verify that the above protocol is indeed an τ_m-OPD protocol for PROMISE-UNION$_m$. But the correctness of the protocol crucially hinges on the fact that the gap for the PROMISE-UNION$_m$ (as defined in 2.12 with parameter $3m/4$ and $m/4$) is very large. Instead, let us define a version of PROMISE-UNION$_m$ with slightly different parameters:

*Definition 6.2 (*NEW-PROMISE-UNION*).* The communication problem NEW-PROMISE-UNION$_m$ is the partial function NEW-PROMISE-UNION$_m : 2^{[m]} \times 2^{[m]} \rightharpoonup \{0,1\}$ defined as

$$\text{NEW-PROMISE-UNION}_m(A,B) = \begin{cases} 1 & \text{if } |A \cup B| \geq 0.6m \\ 0 & \text{if } |A \cup B| \leq 0.4m \end{cases}$$

The proof of Lemma 3.7 goes through to establish an $\Omega(m)$ lower bound for any ρ_m-OPD protocol for NEW-PROMISE-UNION. A lower bound on the communication complexity for τ_m-OPD protocols for NEW-PROMISE-UNION$_m$ would give a lower bound for the space complexity of any ROPD streaming algorithms for estimating F_k for any k.

The ultimate goal of this line of research is to show a lower bound for pseudodeterministic algorithms for F_0 (without the restriction of representation obliviousness). Establishing a lower bound for NEW-PROMISE-UNION against $\mathcal{I}_m$-OPD protocols will yield such a lower bound. Here, $\mathcal{I}_m$ is the trivial identity relation where an input is related to only itself.

In fact, a lower bound for $\mathcal{I}_m$-OPD protocol for NEW-PROMISE-UNION$_m$ is a significant open problem in the area of communication complexity. Only recently, some progress has been made in a similar problem, but in the query complexity regime and not the communication complexity regime [17]. Compared to the problem of obtaining a lower bound for $\mathcal{I}_m$-OPD protocols for the problem of NEW-PROMISE-UNION$_m$, the problem of obtaining lower bounds for τ_m-OPD protocols for NEW-PROMISE-UNION$_m$ seems easie, and proving it would be a natural first step towards proving a lower bound against $\mathcal{I}_m$-OPD protocols.

Acknowledgements

We thank the reviewers for their constructive comments, which helped improve the presentation. Michael Chen and A. Pavan's work is partially supported by NSF grant 2342245 and NSF–DST grant 2413849. Sourav Chakraborty's work is partially supported by DST (Department of Science and Technology, India) grant TPN-104427. N. V. Vinodchandran's work is partially supported by NSF grant 2130608 and NSF–DST grant 2413848.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. 1999. The Space Complexity of Approximating the Frequency Moments. *J. Comput. System Sci.* 58, 1 (1999), 137–147. doi:10.1006/jcss.1997.1545

[2] Nima Anari and Vijay V. Vazirani. 2020. Matching Is as Easy as the Decision Problem, in the NC Model. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 151)*, Thomas Vidick (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 54:1–54:25. doi:10.4230/LIPIcs.ITCS.2020.54

[3] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. 2002. Counting Distinct Elements in a Data Stream. In *Randomization and Approximation Techniques in Computer Science*, José D. P. Rolim and Salil Vadhan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–10.

[4] Vladimir Braverman, Robert Krauthgamer, Aditya Krishnan, and Shay Sapir. 2023. Lower Bounds for Pseudo-Deterministic Counting in a Stream. In *50th International Colloquium on Automata, Languages, and Programming*

(ICALP 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 261), Kousha Etessami, Uriel Feige, and Gabriele Puppis (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 30:1–30:14. doi:10.4230/LIPIcs.ICALP.2023.30

[5] M. Bun, R. Livni, and S. Moran. 2020. An Equivalence Between Private Classification and Online Prediction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, Los Alamitos, CA, USA, 389–402. doi:10.1109/FOCS46700.2020.00044

[6] Jarosław Błasiok. 2018. Optimal Streaming and Tracking Distinct Elements with High Probability. *ACM Transactions on Algorithms (TALG)* 16 (2018), 1 – 28. https://api.semanticscholar.org/CorpusID:4605219

[7] Igor Carboni Oliveira and Rahul Santhanam. 2018. Pseudo-Derandomizing Learning and Approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 116)*, Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 55:1–55:19. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.55

[8] Amit Chakrabarti. 2023. Data Stream Algorithms, Lecture Notes. https://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf

[9] Amit Chakrabarti and Manuel Stoeckl. 2024. Finding Missing Items Requires Strong Forms of Randomness. In *39th Computational Complexity Conference (CCC 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 300)*, Rahul Santhanam (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 28:1–28:20. doi:10.4230/LIPIcs.CCC.2024.28

[10] Sourav Chakraborty, N. V. Vinodchandran, and Kuldeep S. Meel. 2022. Distinct Elements in Streams: An Algorithm for the (Text) Book. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany (LIPIcs, Vol. 244)*, Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 34:1–34:6. doi:10.4230/LIPICS.ESA.2022.34

[11] Peter Dixon, A. Pavan, and N. V. Vinodchandran. 2018. On Pseudodeterministic Approximation Algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 117)*, Igor Potapov, Paul Spirakis, and James Worrell (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 61:1–61:11. doi:10.4230/LIPIcs.MFCS.2018.61

[12] Peter Dixon, A. Pavan, and N. V. Vinodchandran. 2021. Complete Problems for Multi-Pseudodeterministic Computations. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 185)*, James R. Lee (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 66:1–66:16. doi:10.4230/LIPIcs.ITCS.2021.66

[13] Peter Dixon, A. Pavan, Jason Vander Woude, and N. V. Vinodchandran. 2022. Pseudodeterminism: promises and lowerbounds. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (Rome, Italy) *(STOC 2022)*. Association for Computing Machinery, New York, NY, USA, 1552–1565. doi:10.1145/3519935.3520043

[14] Philippe Flajolet and G. Nigel Martin. 1985. Probabilistic Counting Algorithms for Data Base Applications. *J. Comput. Syst. Sci.* 31, 2 (1985), 182–209. doi:10.1016/0022-0000(85)90041-8

[15] Sumit Ganguly. 2008. Lower bounds on frequency estimation of data streams. In *Proceedings of the 3rd International Conference on Computer Science: Theory and Applications* (Moscow, Russia) *(CSR'08)*. Springer-Verlag, Berlin, Heidelberg, 204–215.

[16] Eran Gat and Shafi Goldwasser. 2011. Probabilistic Search Algorithms with Unique Answers and Their Cryptographic Applications. *Electronic Colloquium on Computational Complexity (ECCC)* 18 (01 2011), 136.

[17] Dmytro Gavinsky. 2024. Unambiguous parity-query complexity. *Electron. Colloquium Comput. Complex.* TR24-009 (2024). ECCC:TR24-009 https://eccc.weizmann.ac.il/report/2024/009

[18] Phillip B. Gibbons and Srikanta Tirthapura. 2001. Estimating simple functions on the union of data streams. In *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures* (Crete Island, Greece) *(SPAA '01)*. Association for Computing Machinery, New York, NY, USA, 281–291. doi:10.1145/378580.378687

[19] Oded Goldreich. 2019. Multi-pseudodeterministic algorithms. *Electron. Colloquium Comput. Complex.* TR19-012 (2019). ECCC:TR19-012 https://eccc.weizmann.ac.il/report/2019/012

[20] Oded Goldreich, Shafi Goldwasser, and Dana Ron. 2013. On the possibilities and limitations of pseudodeterministic algorithms. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science - ITCS '13*. ACM Press, Berkeley, California, USA, 127. doi:10.1145/2422436.2422453

[21] Shafi Goldwasser and Ofer Grossman. 2017. Bipartite Perfect Matching in Pseudo-Deterministic NC. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 80)*, Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 87:1–87:13. doi:10.4230/LIPIcs.ICALP.2017.87

[22] Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. 2018. Pseudo-Deterministic Proofs. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 94)*,

Anna R. Karlin (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 17:1–17:18. doi:10.4230/LIPIcs.ITCS.2018.17

[23] Shafi Goldwasser, Ofer Grossman, Sidhanth Mohanty, and David P. Woodruff. 2020. Pseudo-Deterministic Streaming. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 151)*, Thomas Vidick (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 79:1–79:25. doi:10.4230/LIPIcs.ITCS.2020.79

[24] O. Grossman, M. Gupta, and M. Sellke. 2023. Tight Space Lower Bound for Pseudo-Deterministic Approximate Counting. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, Los Alamitos, CA, USA, 1496–1504. doi:10.1109/FOCS57990.2023.00091

[25] Russell Impagliazzo, Rex Lei, Toniann Pitassi, and Jessica Sorrell. 2022. Reproducibility in learning. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (Rome, Italy) *(STOC 2022)*. Association for Computing Machinery, New York, NY, USA, 818–831. doi:10.1145/3519935.3519973

[26] Bala Kalyanasundaram and Georg Schintger. 1992. The Probabilistic Communication Complexity of Set Intersection. *SIAM Journal on Discrete Mathematics* 5, 4 (1992), 545–557. arXiv:https://doi.org/10.1137/0405044 doi:10.1137/0405044

[27] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 2010. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (, Indianapolis, Indiana, USA,) *(PODS '10)*. Association for Computing Machinery, New York, NY, USA, 41–52. doi:10.1145/1807085.1807094

[28] Yi Li, Huy L. Nguyen, and David P. Woodruff. 2014. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing* (New York, New York) *(STOC '14)*. Association for Computing Machinery, New York, NY, USA, 174–183. doi:10.1145/2591796.2591812

[29] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. 2021. Pseudodeterministic algorithms and the structure of probabilistic time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (Virtual, Italy) *(STOC 2021)*. Association for Computing Machinery, New York, NY, USA, 303–316. doi:10.1145/3406325.3451085

[30] Robert Morris. 1978. Counting large numbers of events in small registers. *Commun. ACM* 21, 10 (oct 1978), 840–842. doi:10.1145/359619.359627

[31] Mridul Nandi, N. V. Vinodchandran, Arijit Ghosh, Kuldeep S. Meel, Soumit Pal, and Sourav Chakraborty. 2024. Improved Streaming Algorithm for the Klee's Measure Problem and Generalizations. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2024, August 28-30, 2024, London School of Economics, London, UK (LIPIcs, Vol. 317)*, Amit Kumar and Noga Ron-Zewi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 26:1–26:21. doi:10.4230/LIPICS.APPROX/RANDOM.2024.26

[32] Ilan Newman. 1991. Private vs. common random bits in communication complexity. *Inform. Process. Lett.* 39, 2 (1991), 67–71. doi:10.1016/0020-0190(91)90157-D

[33] Igor C. Oliveira and Rahul Santhanam. 2017. Pseudodeterministic constructions in subexponential time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing* (Montreal, Canada) *(STOC 2017)*. Association for Computing Machinery, New York, NY, USA, 665–677. doi:10.1145/3055399.3055500

[34] A.A. Razborov. 1992. On the distributional complexity of disjointness. *Theoretical Computer Science* 106, 2 (1992), 385–390. doi:10.1016/0304-3975(92)90260-M

Received June 2025; accepted August 2025