Estimation of the Size of Union of Delphic Sets: Achieving Independence from Stream Size*

Kuldeep S. Meel ⓡ
National University of Singapore

Sourav Chakraborty ⓡ
Indian Statistical Institute, Kolkata

N. V. Vinodchandran
University of Nebraska, Lincoln

ABSTRACT

Given a family of sets $\{S_1, S_2, \ldots S_M\}$ over a universe Ω, estimating the size of their union in the data streaming model is a fundamental computational problem with a wide variety of applications. The holy grail in the field of streaming is to seek design of algorithms that achieve (ε, δ)-approximation with $\mathsf{poly}(\log |\Omega|, \varepsilon^{-1}, \log \delta^{-1})$ space and update time complexity.

Earlier investigations achieve algorithms with desired space and update time complexity for restricted cases such as singletons (Distinct Elements problem), one-dimensional ranges, arithmetic progressions, and sub-cubes. However, techniques used in these works fail for many other simple structured sets. A prominent example is that of Klee's Measure Problem (KMP), wherein every set S_i is represented by an axis-parallel rectangle in d-dimensional spaces. Despite extensive prior work, the best-known streaming algorithms for many of these cases depend on the size of the stream, and therefore the problem of whether there exists a streaming algorithm for estimations of size of the union of sets with $\mathsf{poly}(\log |\Omega|, \varepsilon^{-1}, \log \delta^{-1})$ space and update time complexity has remained open.

In this work, we focus on certain general families of sets called *Delphic families* (which allows *efficient* membership, sampling, and cardinality queries). Such families of sets capture several well-known problems, including KMP, test coverage, and hypervolume estimation.

The primary contribution of our work is to resolve the above-mentioned open problem for streams over Delphic families. In particular, we design the first streaming algorithm for estimating $\left|\bigcup_{i=1}^{M} S_i\right|$ with $\mathsf{poly}(\log |\Omega|, \varepsilon^{-1}, \log \delta^{-1})$ space and update time complexity (independent of M, the length of the stream) when each S_i is a member from a Delphic family of sets. We further generalize our results to larger families of sets, called *approximate-Delphic families*, for which the size of a set can be known approximately but not exactly. Our results resolve two of the open problems listed in Meel, Vinodchandran, Chakraborty (PODS-21).

CCS CONCEPTS

• **Theory of computation → Streaming models**; **Sketching and sampling**.

KEYWORDS

probabilistic computations, streaming algorithms, approximation algorithms

ACM Reference Format:
Kuldeep S. Meel ⓡ, Sourav Chakraborty ⓡ, and N. V. Vinodchandran. 2022. Estimation of the Size of Union of Delphic Sets: Achieving Independence from Stream Size. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3517804.3526222

1 INTRODUCTION

The widespread adoption of computing has led to an explosion of data that modern computing systems need to process efficiently. The design of data analysis techniques with computational and storage efficiency is of utmost importance. Consequently, the past two decades have witnessed a sustained interest in the design of efficient streaming algorithms.

In this paper, we focus on one of the fundamental problems in the context of streaming: Given a stream of sets $S_1, S_2, \ldots S_M$, estimate their union $\left|\bigcup_{i=1}^{M} S_i\right|$, which is often referred to as zeroth frequency moment and denoted by F_0 of the stream. The goal, usually, is to design an efficient randomized algorithm that can output an (ε, δ)-approximation of the $|\bigcup_{i=1}^{M} S_i|$. We say that a random variable Z is an (ε, δ)-approximation of Y if $\Pr[|Z - Y| \le \varepsilon |Y|] \ge 1 - \delta$.

We will focus on general families of sets, called Delphic families[1], defined below:

Definition 1.1. *Let Ω be a discrete universe. A set $S \subseteq \Omega$ belongs to a Delphic family if the following queries can be done in $O(\log |\Omega|)$ time.*

Membership *Given any $x \in \Omega$ check if $x \in S$.*
Cardinality *Determine the size of S, i.e. $|S|$.*
Sampling *Draw a uniform random sample from S.*

The goal is to design a streaming algorithm that, given a stream of sets $S_1, S_2, \ldots S_M$ from a Delphic family, computes an (ε, δ)-approximation of $\left|\bigcup_{i=1}^{M} S_i\right|$ while minimising the worst case space complexity and the worst case update time complexity. The (worst case) update time complexity is the (worst case) time spent processing a single item in the stream. In the setting of streams over Delphic sets, it is proportional to the (worst case) number of queries made to a set in the stream. This abstract computational problem over Delphic families captures several well-known problems, including the discrete version of Klee's Measure Problem (KMP), test coverage estimation, hyper-volume estimation, and the DNF counting problem.

*The authors decided to forgo the old convention of alphabetical ordering of authors in favor of a randomized ordering, denoted by ⓡ. The publicly verifiable record of the randomization is available at https://www.aeaweb.org/journals/policies/random-author-order/search

[1]While the name *Delphic sets* was introduced in our prior work [33], the notion of Delphic sets has been implicit in several works over the past three decades [5, 10, 23, 24]; to the best of our knowledge, the first work that explicitly mentions the three properties of Delphic sets is the seminal work of Karp and Luby [23].

PODS '22, June 12–17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9260-0/22/06.
https://doi.org/10.1145/3517804.3526222

The Distinct Elements problem, one of the most widely studied problems in the streaming literature, when cast in terms of set streams, is to estimate the size of the union of sets wherein every set S_i of the stream is a singleton element of the universe. A long line of work culminated in the development of algorithms for Distinct Elements with optimal space complexity $O(\log|\Omega| + \frac{1}{\varepsilon^2})$ and $O(1)$ update time complexity [21] (for a constant δ). The design of algorithms with $\mathrm{poly}(\log|\Omega|, \frac{1}{\varepsilon}, \log\frac{1}{\delta})$ space and update time complexity for the Distinct Elements problem spurred interest in investigations of streaming algorithms for more broader classes of sets. One such example is that of single-dimensional ranges, wherein every S_i is encoded as $[a_i, b_i]$ for $a_i \leq b_i$ and represents the set of all integers x_i such that $a_i \leq x_i \leq b_i$. For this case, the algorithms for distinct elements problem can be used by processing every element of S_i one by one. While such algorithms would provide optimal space complexity $O(\log|\Omega| + \frac{1}{\varepsilon^2})$, the resulting update time complexity of $O(|\Omega|)$ is highly undesirable. Bar-Yossef, Kumar, Sivakumar [3] introduced the notion of *range-efficient* streaming algorithms to capture the desiderata of space and update time complexity to be logarithmic in the size of the range. Subsequently, Pavan and Tirthapura [29], Sun and Poon [31] achieved range-efficient algorithms for single-dimensional ranges.

A natural question is to investigate the design of range-efficient algorithms for *multi-dimensional ranges*, which can also be viewed as a discrete version of the well-known Klee's Measure Problem (KMP) [32, 34]. KMP is a natural and fundamental problem studied in computational geometry resulting in a substantial body of research. [4–8, 14, 17, 25, 28] [2]. KMP also arises naturally in spatial databases [33, 34]. Furthermore, a restricted variant of KMP, known as the *Hyper Volume Estimation* problem, is an important computational problem studied in evolutionary algorithms [5]. Initial attempts to design range-efficient algorithms for KMP in the streaming setting, however, failed to achieve $\mathrm{poly}(\log|\Omega|, \varepsilon^{-1}, \log\delta^{-1})$ space and update time complexity. In particular, such attempts yielded techniques with update time complexity of $O(|\Omega|)$ [32, 34].

Recently, Meel, Vinodchandran, and Chakraborty [33] took a promising step and achieved $\mathrm{poly}(\log|\Omega|, \varepsilon^{-1}, \log\delta^{-1}, \log M)$ space and update time complexity for KMP and more generally for computing F_0 of a stream of Delphic sets of stream-size M. However, the scheme due to MVC[3] still falls short of desiderata (of obtaining a space and update time complexity of $\mathrm{poly}(\log|\Omega|, \varepsilon^{-1}, \log\delta^{-1})$ and independent of the stream size) in streaming literature given its dependence on the size of the stream. To summarize, despite extensive prior work, design of algorithms with $\mathrm{poly}(\log|\Omega|, \varepsilon^{-1}, \log\delta^{-1})$ space and update time complexity for set streams over Delphic families remains open and is of significant interest from theoretical and practical perspectives.

1.1 Our Results

The primary contribution of this work is to resolve the above-mentioned open problem. In particular, we design the first streaming algorithm for estimation of $\left|\bigcup_{i=1}^{M} S_i\right|$ with $\mathrm{poly}(\log|\Omega|, \frac{1}{\varepsilon}, \log\frac{1}{\delta})$ space and update time complexity (independent of M, the length of the stream). Formally, we prove the following theorem:

Theorem 1.2. *There is a streaming algorithm, which we call* Vatic, *that given real numbers $\varepsilon, \delta < 1$, and a stream $\mathcal{S} = \langle S_1, S_2, \cdots, S_M\rangle$ of unknown length M where each $S_i \subseteq \Omega$ belongs to a Delphic family, computes an (ε, δ)-approximation of $\left|\bigcup_{i=1}^{M} S_i\right|$.*

The algorithm has the worst case space complexity $O(\log^3(|\Omega|) \cdot \frac{\log(1/\delta)}{\varepsilon^2})$ and the update time complexity $\widetilde{O}(\log^4(|\Omega|) \cdot \frac{\log(1/\delta)}{\varepsilon^2})$.

As a corollary to Theorem 1.2, we get an algorithm with space and update time complexity $\mathrm{poly}(\log|\Omega|, \frac{1}{\varepsilon}, \log\frac{1}{\delta})$ for Klee's Measure Problem, which is formally defined in Definition 2.2.

Corollary 1.3. *There is a streaming algorithm that given real numbers $\varepsilon, \delta < 1$, and a stream $\mathcal{R} = \langle \boldsymbol{r}_1, \boldsymbol{r}_2, \cdots \boldsymbol{r}_M\rangle$ where each $\boldsymbol{r}_i$ is a d-dimensional rectangle over $\Omega = \Delta^d$, computes an (ε, δ)-approximation of* $\mathrm{Volume}(\mathcal{R})$. *The worst case space complexity of the algorithm is $O(d^3 \log^3(|\Delta|) \cdot \frac{\log(1/\delta)}{\varepsilon^2})$, while its update time complexity is $\widetilde{O}(d^4 \log^4(|\Delta|) \cdot \frac{\log(1/\delta)}{\varepsilon^2})$.*

While the framework of the Delphic set is general enough to capture many important scenarios, there are settings where it is impossible to obtain the size of a set exactly. Similarly, getting a sample uniformly at random from a set can also be challenging. To handle the problem of estimating the size of the union of such sets, we consider a natural generalization of the notion of Delphic Families called *Approximate-Delphic Families.*

Definition 1.4. *Let Ω be a discrete universe. A set $S \subseteq \Omega$ belongs to an Approximate-Delphic family if for some constants $0 \leq \alpha, \gamma, \eta$ there is an oracle that allows the following set of queries.*

Membership *Given any x check if $x \in S$.*

Approximate Cardinality *Get an approximation of the size of S which with probability $\geq (1-\gamma)$ is between $|S|/(1+\alpha)$ and $(1+\alpha)|S|$. We call such an approximation (α, γ)-approximation of $|S|$.*

Approximate Sampling *Draw a random sample from S where the probability that any element $x \in S$ is sampled is between $\frac{1}{(1+\eta)|S|}$ and $\frac{(1+\eta)}{|S|}$. We call such as oracle η-random sampling oracle.*

We will refer to such an oracle as an (α, γ, η)-Approximate-Delphic oracle [4].

Several families of sets such as convex sets, star-shaped sets, and Schlicht Domains (see Section 6.2) fall under the category of Approximate-Delphic families. Thus a streaming algorithm for estimating the union of sets when given access to an (α, γ, η)-Approximate-Delphic oracle gives a streaming algorithm for estimating the union of sets for the aforementioned families of sets.

Our next result is an algorithm that can approximate the size of the union of sets given access to an (α, γ, η)-Approximate-Delphic Oracle.

Theorem 1.5. *There is a streaming algorithm, which we call* Ext-Vatic *that, given real numbers $\varepsilon, \delta < 1$, and a stream $\mathcal{S} = \langle S_1, S_2, \cdots, S_M\rangle$ of unknown length M where each $S_i \subseteq \Omega$ belongs to an Approximate-Delphic family, and access to an (α, γ, η)-Approximate-Delphic oracle for some α, γ, δ for members of the family, outputs a number in the range $[\frac{(1-\varepsilon)}{2(1+\eta)(1+\alpha)}|\bigcup_{i=1}^{M} S_i|, (1+\varepsilon)(1+\eta)(1+$*

[2]The formal definition on KMP is given in Definition 2.2.

[3]Named after the initials of authors.

[4]The reason for using the above notions of approximation is discussed in Section 2.

$\alpha)|\bigcup_{i=1}^{M} S_i|]$ *The worst case space complexity of the algorithm is* $O((\log^3|\Omega|)\log(1/\delta)\cdot\frac{(1+\eta)}{\epsilon^2})$. *The algorithm, while processing any item of the stream, makes*

$$\tilde{O}((\log^3|\Omega|)\log(1/\delta)\log(\frac{1}{1-\gamma})\frac{(1+\eta)}{\epsilon^2})$$

calls to the (α,γ,η)*-Approximate-Delphic Oracle in the worst case.*

The techniques that we use to extend Vatic to Ext-Vatic can also be used to extend the streaming algorithm for Delphic sets in [33] to handle Approximate-Delphic sets. This addresses a problem that was left open in [33]. The extension of their algorithm to Approximate-Delphic sets is presented in Appendix D.

Remark 1.6. *In the definition of the Delphic family, we do not make any restrictions about the representations of sets. Instead, we assume that the streaming algorithm gets a set S in some representation on the input memory. The resource requirements are not explicitly parameterized by this representation but rather by the size of the universe of the set S. This allows us to state the resource requirements of our algorithm in line with those in the literature. Moreover, the applications we state fit this model naturally. However, in the definition of the Approximate-Delphic family, we use the oracle formulation where each operation takes a unit time step. This is because for applications we present in this paper, these operations can be non-trivial and known algorithms take polynomial time in the standard representations.*

We also note here that for a family of sets for which there is an efficient algorithm for membership testing given some representation, there is also a succinct representation for every element in the family in the form of Boolean circuits. To state this succinct representation theorem, we consider the universe of Boolean strings. This is without loss of generality, as any universe Ω *can be encoded in* $\{0,1\}^{\lceil\log|\Omega|\rceil}$. *Let* $\{\mathcal{F}_n\}_n$ *be a series indexed by integers* $n \geq 1$ *where each* $\mathcal{F}_n$ *is a family of sets in* $\{0,1\}^n$ *with a set of representations* $\mathcal{R}$*: that is for any n, any* $S_n \in \mathcal{F}_n$ *is represented by some* $R_{S_n} \in \mathcal{R}$. *For a set* $S \subseteq \{0,1\}^n$, *we say that a Boolean circuit C on n inputs represents S if for all* $x \in \{0,1\}^n$, $x \in S$ *if an only if* $C(x)$ *evaluates to True. The following is a well-known theorem from Computational Complexity. The Boolean circuits we consider are the ones with fan-in 2 AND and OR gates and fan-in 1 NOT gates. The* size *of a circuit is the number of gates in it.*

Theorem 1.7 ([9, 19]). *If there is a membership testing algorithm that on input* $\langle R_{S_n}, x\rangle$, *outputs 'Yes' if* $x \in S_n$ *and 'No' if* $x \notin S_n$ *in time* $T(n)$ *where* $n = |x|$, *then for all n, there is a Boolean circuit* C_n *of size* $O(T(n)\log T(n))$ *that represents* S_n. *In particular if there is a membership testing algorithm that runs in time* $O(n)$, *then there is a Boolean circuit of size* $\tilde{O}(n)$ *that represents the set* S_n *for every member of the family.*

1.2 Our Techniques

Our algorithm is based on a simple but general sampling-based strategy. Let $\bigcup_i S_i = \{s_1, s_2, \ldots, s_k\} \subseteq \Omega$, where $k = |\bigcup_i S_i|$. The main idea is to sample each s_j independently with appropriately chosen probability p_j and store the tuple (s_j, p_j): the element along with the probability with which it was sampled, in a bucket $\mathcal{X}$. At the end of the stream, we can compute our estimate $\sum_j \frac{N(p_j)}{p_j}$[5] where $N(p_j)$ represents the number of elements in $\mathcal{X}$ that were sampled with probability p_j. Our objective is to obtain an algorithm with poly$(\log|\Omega|, \varepsilon^{-1}, \log\delta^{-1})$ space and update time complexity; therefore, the size of $\mathcal{X}$ is expected to be of the same order of magnitude; in particular, we will maintain $|\mathcal{X}| \in O(\log^2|\Omega|\cdot\frac{\log\delta^{-1}}{\varepsilon^2})$.

There are two key challenges we need to overcome: (C1) there may be many sets S_i such that $s_j \in S_i$ and (C2) how do we choose value p_j.

To address the challenge C1, we borrow the simple but powerful technique first introduced in [33]: when processing S_i, remove all elements from $\mathcal{X}$ that lie in S_i. Therefore, whether $s_j \in \mathcal{X}$ depends only on the last occurrence of s_j, i.e., the last set S_i for which $s_j \in S_i$

We now turn to the most critical challenge, C2. To this end, we first note that the estimate $\sum_j \frac{N(p_j)}{p_j}$ is an unbiased estimator of $\left|\bigcup_{i=1}^{M} S_i\right|$. Since we sample each s_i independently, the standard concentration bounds would yield (ε,δ)-guarantees as long as every element is sampled with *sufficiently high* probability. Observe that when the elements are sampled with a very small probability, then central moments of the estimator are too high in comparison to the expectation. Technically, it suffices to have $p_j \geq \frac{1}{k}$. However, there is, no apriori *good* estimate of k; our problem is, after all, to estimate k. One possible strategy, explored in [33], would be to start with setting $p = 1$ and decrease p every time the bucket $\mathcal{X}$ reaches its capacity. To ensure that every element s_j is picked with $p_j \geq \frac{1}{k}$ (with high probability), we would have to ensure that at every point of the stream of length M, the value p does not fall below $\frac{1}{k}$, which leads to a $\log M$ factor in the performance. Our key insight is that an element s_j need not be picked with probability $p \geq \frac{1}{k}$ whenever s_j occurs in the stream, as whether $s_j \in \mathcal{X}$ depends only on the last occurrence of s_j. Therefore, we only need to ensure that the last time s_j appears, it should be picked with probability $p_j \geq \frac{1}{k}$. A potential obstacle is that it is not possible to determine if s_j will occur in the future or not. We resolve the issue by observing that if we decide on the probability p with which elements of S_i should be picked based on the size of $\mathcal{X}$, then we can lower bound the probability p_j for each s_j without any assumptions on the stream. We give some details. Let $\mathcal{I} \subseteq [M]$ be the set of indices corresponding to the last occurrences for s_j's. Formally, $i \in \mathcal{I}$ if for some s_j, we have $s_j \in S_i$ and there is no $i' > i$ such that $s_j \in S_{i'}$. Observe that regardless of the value of M, since $|\bigcup_i S_i| \leq |\Omega|$ and there is a surjection between $\bigcup_i S_i$ and $\mathcal{I}$, we have $|\mathcal{I}| \leq |\Omega|$. Therefore, to bound the probability that for all s_j, we have $p_j \geq \frac{1}{k}$, we need to perform union bound over at most $|\Omega|$ events, thereby, leading to a $\log|\Omega|$ factor in the expression for $|\mathcal{X}|$. It is worth emphasizing that we do not seek to process every occurrence of s_j with $p \geq \frac{1}{k}$ and therefore, we allow for the possibility that except for the last occurrence, s_j was sampled with probability less than $\frac{1}{k}$.

Organization: The rest of the paper is organized as follows. We discuss notations and preliminaries in Section 2. In Section 3 we describe related works. In Section 4, we present our main algorithm Vatic and prove its correctness and establish complexity bounds, thus proving Theorem 1.2. In Section 5, we present Ext-Vatic (an

[5]For technical reasons, the estimator in our algorithm involves further resampling step.

extension of VATIC) that works for set streams over Approximate-Delphic families. Finally, in Section 6 we present a number of applications of our algorithms.

2 NOTATIONS AND PRELIMINARIES

We will denote by $[n]$ the set of natural numbers $\{1, 2, \ldots, n\}$ and by $\binom{[n]}{t}$ the set of all subsets of $[n]$ of size t. For any $t \in \mathbb{N}$ and any $p \in [0,1]$ we will also use $\mathrm{Bin}(t,p)$ to denote the binomial distribution over the set $[t]$ where probability of a number $0 \leq m \leq t$ is $\binom{t}{m} p^m (1-p)^{t-m}$.

The main computational problem is the following.

Definition 2.1 (Estimating the union of (Approximate) Delphic Sets)**.** *Given a stream of sets $S_1, S_2, \ldots S_M$ where each S_i is from an (Approximate) Delphic family, give an (ε, δ)-approximation of the union $\left|\bigcup_{i=1}^{M} S_i\right|$.*

An important and well studied instantiation of the above generic problem is the streaming version of the Klee's Measure Problem (KMP). In the following definition Δ could be any totally ordered set, but without loss of generality we assume $\Delta = [n]$ for some n.

Definition 2.2. *A d-dimensional axis aligned rectangle $\boldsymbol{r}$ over the universe $\Omega = \Delta^d$ is defined as a set $[a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$, where $\forall i$, $a_i, b_i \in \Delta$ and $a_i \leq b_i$. Given a rectangle $\boldsymbol{r}$, let $\mathsf{Range}(\boldsymbol{r})$ denote the set of tuples $\{(x_1, \ldots, x_d)\}$ where $a_i \leq x_i \leq b_i$ and $x_i \in \Delta$. For a set of rectangles $\mathcal{R} = \{\boldsymbol{r}_1, \cdots \boldsymbol{r}_M\}$, the volume of $\mathcal{R}$ is defined as*

$$\mathsf{Volume}(\mathcal{R}) = | \cup_{1 \leq i \leq M} \mathsf{Range}(\boldsymbol{r}_i)|$$

Definition 2.3 (Streaming KMP)**.** *Given ε, δ, and a stream $\mathcal{R} = \langle \boldsymbol{r}_1, \boldsymbol{r}_2, \cdots \boldsymbol{r}_M \rangle$, where each item $\boldsymbol{r}_i$ is a d-dimensional rectangle over Δ^d, compute a (ε, δ)-approximation* $\mathsf{Volume}(\mathcal{R})$.

Note that every d-dimensional rectangle can be naturally and succinctly represented by the tuple $(a_1, b_1, \cdots a_d, b_d)$. KMP is an instantiation of the general framework since every rectangle $\mathbf{r}_i$, defines a set $S_i = \mathsf{Range}(\mathbf{r}_i)$, that satisfies the desired properties of Delphic sets (see [33] for a proof) and $\mathsf{Volume}(\mathcal{R}) = \left|\bigcup_{i=1}^{M} S_i\right|$.

As done in the case of traditional space bounded computations, for counting space, we will not include the space required to represent the input item. We will consider that input is available on a read-only input tape (with random access) and do not contribute to the space used by the algorithm. We consider unit-cost model and assume all basic operations including arithmetic operations on words can be performed in unit time. When the sets are Delphic then, from the definition of Delphic sets, we know that the time complexity for one query is $O(\log|\Omega|)$. So for the Delphic sets the update time complexity (or the time complexity for processing an item in the stream) will turn out to be $O(\log|\Omega|)$ times the number of oracle queries made while processing an item in the stream.

Notions of Approximations: We use two notions of (multiplicative) approximation of a number. When we are concerned with approximation algorithms for the size of the union of sets in a stream (as in Theorem 1.2) our goal is to design an randomized algorithm that is a (ε, δ)-approximation of the size of the union of the sets, where a random variable Z (output of the algorithm) is an (ε, δ)-approximation of Y if $\Pr[|Z - Y| \leq \varepsilon|Y|] \geq 1 - \delta$. In particular, we assume $\varepsilon < 1$. A weaker notion of approximation is used in the definition of Approximate-Delphic oracles. A call to an Approximate-Delphic oracle (Definition 1.4) for cardinality of the set is required to return an (α, δ)-approximation of the size of a set S, where a random variable Z is an (α, δ)-approximation of $|S|$ if $\Pr[\frac{|S|}{(1+\alpha)} \leq Z \leq (1+\alpha)|S|] \geq 1 - \delta$. Note that the second notion of approximation is weaker (less demanding) than the first notion of approximation. In particular, we allow the approximation parameter α to be greater than 1. Thus we design algorithms that approximates the size of the union of sets using the stronger notion of approximation, while when designing algorithms for set streams over Approximate-Delphic families the algorithm can work with queries that gives a weaker guarantee in the approximation of the size of a set. It will be clear from the context which notion of approximation is being referred to.

THEOREM 2.4 (COUPON COLLECTOR PROBLEM). *Given access to uniform random samples from a set T and a number $r \leq |T|$, let Z_r be a random variable that stands for the number of independent uniform random samples from T needed before we get r distinct samples from T. Then for any $\beta \geq 1$,*

$$\Pr[Z_r > \beta r \log r] \leq r^{-(\beta/2)+1}.$$

The proof of Theorem 2.4 for the case when $|T| = r$ is presented in [27]. For completeness we present the proof of Theorem 2.4 in the Appendix. We note that the upper bound can be improved to $r^{-\beta+1}$ with a more involved proof. However, for our purposes the weaker bound suffices.

Independently picking elements from a set with a fixed probability. A crucial operation that was used in [33] for their streaming algorithm for Delphic sets is to sample a subset $\mathcal{L}$ of a set S so that each element of S is in $\mathcal{L}$ independently with probability p for a given probability value p. This operation is implemented by the following sampling process $\mathcal{P}$: *first draw a number K according to the Binomial distribution $B(|S|, p)$ and then draw K distinct elements at random from S.* We will also use this operation in our algorithm. For completeness we give the proof of correctness of this process below.

Claim 2.5. *The sampling process $\mathcal{P}$ samples each element of S independently with probability p.*

The proof of Claim 2.5 is presented in the Appendix.

3 RELATED WORK

Karp and Luby [23] considered the problem of determining the cardinality of union of Delphic sets. Their setting assumed storage of the entire stream, and the resulting algorithms are quite unfriendly to streaming setting. In particular, a straightforward adaption of Karp and Luby [23] (and the subsequent work of Karp, Luby, and Madras [24]) would yield an algorithm with space and time complexity $O(\frac{M \log|\Omega|}{\epsilon^2} \log M \log n)$; the linear dependence on M is highly undesirable from a streaming perspective.

A significant breakthrough for union of sets in streaming setting is due to Flajolet–Martin [13], who focused on the restricted case of singleton sets, also known as Distinct Elements problem. Flajolet–Martin's proposed scheme had, however, assumed access to hash functions with strong independence. This independence requirement was relaxed in the seminal work of Alon, Matias, and Szegedy [1], who demonstrated that pairwise independent hash functions suffice in the context of Distinct Elements. Alon, Matias, and Szegedy kick started a long line of work on streaming algorithms and Distinct Elements in particular, which culminated in the design of algorithms with optimal space complexity $O(\log|\Omega| + \frac{1}{\varepsilon^2})$ and $O(1)$ update time [2, 15, 21].

Spurred by the success of design of algorithms with space complexity independent of M and with logarithmic dependence on $\log|\Omega|$ in the context of Distinct Elements problem, subsequent work sought to handle broader classes of sets; of which a large body of work can be categorized under the category of *range-efficient* algorithms owing to the initial focus on the cases wherein every S_i represents a range $[a_i, b_i]$ i.e., all x such that $a_i \leq x \leq b_i$. As noted earlier, Pavan and Tirthapura [29], Sun and Poon [31] achieved *range-efficient* algorithms for single-dimensional ranges, which is special of KMP for one dimension. The success in attempts to achieve range-efficient algorithms for general version of the problem was limited in the following years. In particular, Thirthapura and Woodruff [34] achieved an algorithm with optimal space complexity but the update time of the algorithm was $O(|\Omega|)$. Subsequently, Pavan, Vinodchandran, Bhattacharyya, and Meel [32] also proposed another hashing-based technique with worst case time complexity of $O(|\Omega|)$.

The state of affairs was recently improved by Meel, Vinodchandran, and Chakraborty [33] who designed a sampling-based strategy that yielded the first algorithm with $\mathrm{poly}(\log|\Omega|, \varepsilon^{-1}, \log\delta^{-1}, \log M)$ space and update time complexity. In this context, it is worth remarking that while the scheme due to Meel, Vinodchandran, and Chakraborty shares high-level similarities with our algorithm; there are crucial technical differences. In particular, their focus is to ensure that every item of the stream is sampled with $p \geq \frac{1}{k}$, where $k = |\cup_i S_i|$, which yields a dependence of M; while we do take a different route, as described in Section 1.2, to achieve bounds independent of the stream size.

4 VATIC: AN ALGORITHM FOR UNKNOWN STREAM SIZE

In this section we prove the following theorem.

Theorem 1.2. *There is a streaming algorithm, which we call* Vatic, *that given real numbers $\varepsilon, \delta < 1$, and a stream $\mathcal{S} = \langle S_1, S_2, \cdots, S_M\rangle$ of unknown length M where each $S_i \subseteq \Omega$ belongs to a Delphic family, computes an (ε, δ)-approximation of $\left|\bigcup_{i=1}^{M} S_i\right|$.*

The algorithm has the worst case space complexity $O(\log^3(|\Omega|) \cdot \frac{\log(1/\delta)}{\varepsilon^2})$ and the update time complexity $\widetilde{O}(\log^4(|\Omega|) \cdot \frac{\log(1/\delta)}{\varepsilon^2})$.

The algorithm, which we call Vatic, maintains a set $\mathcal{X}$ of tuples (s, p) where $s \in \Omega$ and $0 < p \leq 1$ is a probability value, which is initialized to the empty set in the beginning. Each set of the stream is processed by the outer **for** loop (*lines* 3 - 17). At the i^{th} iteration when the set S_i arrives, the algorithm first removes all elements from $\mathcal{X}$ that are in S_i (*lines* 4-6). Then it sets the 'correct' sampling rate p for the set S_i (*lines* 7- 10). During this computation, it also generates a number N_i according to the Binomial distribution $\mathrm{Bin}(|S_i|, p)$. The algorithm proceeds if $p \geq \frac{\log(4/\delta)}{\varepsilon^2|\Omega|}$ and independently samples N_i distinct elements from S_i and adds to $\mathcal{X}$ (*lines* 12 - 17). Since the Delphic sets framework only allows sampling with replacement, in order to sample N_i distinct elements, the algorithm generates up to K_i samples for an appropriate value K_i (set so that by Coupon Collector bound we can guarantee N_i distinct elements are drawn with high probability). Finally, after all the elements in the stream are processed, the algorithm updates $\mathcal{X}$ so that every element is present in $\mathcal{X}$ with the lowest probability p_0 among all sampling probabilities (*lines* 18 - 20).

Algorithm 1 Vatic

1: Initialize $B \leftarrow 6 \cdot \left(\frac{\log(4/\delta)}{\varepsilon^2} \log\left(\frac{4|\Omega|}{\delta}\right)\right)$
2: Initialize $\mathcal{X} \leftarrow \emptyset$
3: **for** $i = 1$ to M **do**
4:     **for** all $(s, *) \in \mathcal{X}$ **do**
5:         **if** $s \in S_i$ **then**
6:             remove $(s, *)$ from $\mathcal{X}$
7:     Set $p \leftarrow 1/2^{\lceil |\mathcal{X}|/B \rceil}$
8:     $N_i \leftarrow \mathrm{Bin}(|S_i|, p)$
9:     **while** $p > 1/2^{\lceil (|\mathcal{X}|+N_i)/B \rceil}$ and $p \geq \frac{\log(4/\delta)}{\varepsilon^2|\Omega|}$ **do**
10:         $N_i \leftarrow \mathrm{Bin}(N_i, 1/2)$ and $p \leftarrow p/2$
11:     **if** $p \geq \frac{\log(4/\delta)}{\varepsilon^2|\Omega|}$ **then**
12:         Set $K_i \leftarrow 4N_i \cdot \log(\frac{4\Omega}{\delta})$; $\quad \mathcal{L} \leftarrow \emptyset$
13:         **for** k = 1 to K_i **do**
14:             $y \leftarrow \mathsf{Sample}(S_i)$
15:             **if** $|\mathcal{L}| < N_i$ **then**
16:                 $\mathcal{L} \leftarrow \mathcal{L} \cup \{(y, p)\}$
17:         $\mathcal{X} \leftarrow \mathcal{X} \cup \mathcal{L}$;
18: Let $p_0 = \min\{p_s \mid \exists s, \ (s, p_s) \in \mathcal{X}\}$
19: **for** $(s, p_s) \in \mathcal{X}$ **do**
20:     With probability $(1 - p_0/p_s)$ remove (s, p_s) from $\mathcal{X}$
21: Estimator: **return** $\frac{|\mathcal{X}|}{p_0}$

Proof. We will now prove the correctness guarantee of Vatic. To this end, we first prove that with high probability every element y in $\bigcup_{i=1}^{m} S_i$ is sampled with probability at least $\frac{\log(4/\delta)}{\varepsilon^2 \cdot |\Omega|}$. A crucial observation is that, since before processing any set S, we remove all the elements of $S \cap \mathcal{X}$ from $\mathcal{X}$, the event '$y \in \mathcal{X}$' only depends on the outcome of sampling from the *last set* in which y is present.

We fix an arbitrary $y \in \bigcup_{i=1}^{m} S_i$. We first define an event Good as follows. For a element y, let S_j be the last set in the stream where $y \in S_j$ and let p_y be the random variable that dictates the probability with which the elements of S_j are sampled and added to $\mathcal{X}$. Let $D = 2^{\left\lceil \log\left(\frac{\log(4/\delta)}{\varepsilon^2 \cdot |\cup_{i=1}^{j} S_i|}\right)\right\rceil}$. Note that since the range of values taken by p_y is a (negative) power of 2, the event '$p_y < D$' and the event '$p_y < \frac{\log(4/\delta)}{\varepsilon^2 \cdot |\cup_{i=1}^{j} S_i|}$' are identical. Let F_y be the event that '$p_y < D$.' Then the event Good is defined as: Good $= \overline{\bigcup_{y \in \cup_{i=1}^{M} S_i} F_y}$ (the complement of $\bigcup_{y \in \cup_{i=1}^{M} S_i} F_y$).

We first prove the following claim.

Claim 4.1. $\Pr[\mathrm{Good}] \geq 1 - \frac{\delta}{2}$

Proof. Let $\mathcal{X}_j$ represent the set $\mathcal{X}$ at *line* 3 when $i = j$. First, observe that for the event F_y to happen, one of the following events should happen: (C1) at the end of the **while** loop 7– 10, we have $p < D$; we will denote this event as F_y^1, or (C2) we fail to sample at least N_j distinct elements in the **for** loop 13– 16; we will denote this event as F_y^2. This is because if we sample N_j distinct elements from S_j where $N_j \sim \text{Bin}(|S_j|, p)$, then by Claim 2.5, every element of S_j will be independently sampled with probability p. Therefore the event that elements of S_j are sampled with probability $< p$ implies the event $< N_j$ samples are chosen.

Therefore, $\Pr[F_y] \leq \Pr[F_y^1 \cup F_y^2]$. We will now upper bound both F_y^1 and F_y^2.

Bounding the probability of F_y^1: Let $N_j(D)$ denote the value of N_j when $p = D$ in *line* 9. For F_y^1 to happen, it must be the case that $\lceil(|\mathcal{X}_j| + N_j(D))/B\rceil > \log(1/D)$, which implies that

$$\mathcal{X}_j + N_j(D) \geq B \cdot \log\left(\frac{1}{2D}\right) \tag{1}$$

Now observe that for every iteration k of the outer **for** loop 3– 17, for all (s, p_s) tuples added to $\mathcal{X}$, it holds true that $p_s < 1/2^{\lceil(|\mathcal{X}_{k+1}|)/B\rceil}$ (recall, $\mathcal{X}_{k+1}$ denotes the set $\mathcal{X}$ at line 3 when $i = k+1$; i.e., after the end of the iteration k). In other words, during the entire run of the algorithm, a tuple (s, p_s) will not be added to $\mathcal{X}$ whenever $|\mathcal{X}| > B \cdot \lfloor\log(1/p_s)\rfloor$. Therefore, the following invariant holds true in the entire run of the algorithm:

$$|\{(s, p_s) \in \mathcal{X} \mid p_s \geq \ell\}| \leq B \cdot \lfloor\log 1/\ell\rfloor \tag{2}$$

Substituting $\ell = 4D$ and observing $\lfloor\log\left(\frac{1}{4D}\right)\rfloor = \log\left(\frac{1}{4D}\right)$, in Eq 2, we have

$$\left|\left\{(s, p_s) \in \mathcal{X}_j \mid p_s \geq 4D\right\}\right| \leq B \cdot \log\left(\frac{1}{4D}\right). \tag{3}$$

Combining Eq 1 and Eq 3, we have

$$\left|\left\{(s, p_s) \in \mathcal{X}_j \mid p_s \leq 2D\right\}\right| + N_j(D) \geq B\log\left(\frac{1}{2D}\right) - B\log\left(\frac{1}{4D}\right) = B$$

Let us define a random variable $Z_j(p)$ to denote the size of set obtained by picking every element of $|\cup_{i=1}^{j} S_i|$ independently with probability p. Based on Chernoff Bound, we have $\Pr[Z_j(2D) \geq B] \leq \frac{\delta}{4|\Omega|}$. Therefore,

$$\begin{aligned}\Pr[F_y^1] &\leq \Pr[\left|\left\{(s, p_s) \in \mathcal{X}_j \mid p_s < 2D\right\}\right| + N_j(D) \geq B] \\ &\leq \Pr[Z_j(2D) \geq B] \leq \frac{\delta}{4|\Omega|}\end{aligned}$$

Bounding the probability of F_y^2: To this end, observe that from the Coupon Collector Theorem 2.4, we can bound $\Pr[|\mathcal{L}| < N_i] \leq \frac{\delta}{4|\Omega|}$. Therefore, we have $\Pr[F_y] \leq \Pr[F_y^1] + \Pr[F_y^2] \leq \frac{\delta}{2|\Omega|}$. Finally, by observing that $\frac{\log(4/\delta)}{\varepsilon^2 \cdot \left|\bigcup_{i=1}^{M} S_i\right|} \geq \frac{\log(4/\delta)}{\varepsilon^2 \cdot |\cup_{i=1}^{j} S_i|}$ for all j and taking union bound over all F_y, we obtain our desired probability. □

Now, we are ready to prove the correctness guarantee of Vatic. To this end, we first observe that the expected value of the output of the algorithm, $\mathbb{E}\left(\frac{|\mathcal{X}|}{p_0} \mid \text{Good}\right) = \left|\bigcup_{i=1}^{M} S_i\right|$.

Let us denote the event that 'the output of Vatic is outside the interval $[(1-\varepsilon)\left|\bigcup_{i=1}^{M} S_i\right|, (1+\varepsilon)\left|\bigcup_{i=1}^{M} S_i\right|]$' by Error. Then, we can bound $\Pr[\text{Error} \mid \text{Good}]$ by a straightforward application of Chernoff bound.

$$\begin{aligned}\Pr[\text{Error} \mid \text{Good}] &= \Pr\left[\left|\frac{|\mathcal{X}|}{p_0} - \left|\bigcup_{i=1}^{M} S_i\right|\right| \geq \varepsilon\left|\bigcup_{i=1}^{M} S_i\right| \mid \text{Good}\right] \\ &\leq \delta/2\end{aligned}$$

Hence, $\Pr[\text{Error}] \leq \Pr[\overline{\text{Good}}] + \Pr[\text{Error} \mid \text{Good}] \leq \frac{\delta}{2} + \frac{\delta}{2} = \delta$.

Correctness of the space complexity bound: From the invariant as stated in Eq 2 and the bound that $p_0 \geq \frac{\log(4/\delta)}{\varepsilon^2 \cdot \left|\bigcup_{i=1}^{M} S_i\right|} \geq 1/|\Omega|$, we have that at any point of the execution of the algorithm, $|\mathcal{X}| \leq \log|\Omega| \cdot B = O(\log^2|\Omega| \cdot \frac{\log(1/\delta)}{\varepsilon^2})$. An element of $\mathcal{X}$ takes $O(\log(|\Omega|))$ space to store. Hence the space complexity is $O(\log^3|\Omega| \cdot \frac{\log(1/\delta)}{\varepsilon^2})$.

Correctness of the update time bound: Note that for processing a set S_i, the time to sample N_i distinct elements from S_i (from *lines* 13 to 16) dominates the rest of the running time, which is invoked at most K_i times. Therefore, since each sampling operation takes $O(\log\Omega)$, the total update time is $\widetilde{O}(\log^4(|\Omega|) \cdot \frac{\log(1/\delta)}{\varepsilon^2})$.

□

5 APPROXIMATE-DELPHIC SETS

We begin by making a few observations about (α, γ, η)-Approximate-Delphic Oracles. The first observation is that the probability of success of the oracle call for the approximate cardinality of a set can be amplified using the median trick (by making multiple queries and outputting the median value) - the proof follows from a standard application of Chernoff's bounds. The second observation is on getting K distinct samples from a set using the approximate sampling oracle. The proof of the second item follows from the bound on the Coupon Collector problem.

Observation 5.1. *(1) Given access to an Approximate-Delphic set S through the (α, γ, η)-Approximate-Delphic oracle that gives an (α, γ)-approximation of $|S|$, by querying the oracle $O(\log T\gamma)$ times we can obtain an $(\alpha, 1/T)$-approximation of $|S|$, for any integer T. Also, if K is an $(\alpha, 1/T)$-approximation of $|S|$ then $(1+\alpha)K$ has the guarantee that with probability $\geq (1 - 1/T)$*

$$|S| \leq (1+\alpha)K \leq (1+\alpha)^2|S|.$$

(2) Given access to a set S through the (α, γ, η)-Approximate-Delphic Oracle, for any K, using $O((1+\eta)K\log(TK))$ samples from (α, γ, η)-Approximate-Delphic oracle to sample from S, with probability $\geq (1 - 1/T)$ we can obtain at least $\min\{K, |S|\}$ distinct samples of S. In particular, for the case $K = |S|$, with $O((1+\eta)|S|\log(T|S|))$ approximate sampling oracle queries, we can compute $|S|$ with probability $\geq (1 - 1/T)$.

The algorithm for Approximate-Delphic families follows the approach of Vatic. But before we present the algorithm, we need to make some crucial observations about the implementation of Vatic and how to adapt it to work for a set stream over Approximate-Delphic family.

A crucial operation that we use for the implementation of our algorithm Vatic is that drawing each element of a set S independently with probability p for a fixed probability p. Claim 2.5 shows that this can be implemented by sampling process $\mathcal{P}$: first by drawing a number K according to the Binomial distribution $B(|S|, p)$ and then drawing K distinct elements at random from S.

The above process crucially depends on knowing the exact size of the set S and that one can sample uniformly at random from the set S. These are not guaranteed in the case of Approximate-Delphic sets. However, we argue that we can work with approximations to implement the sampling procedure.

First, let us assume that we have $|S|$ but we only have access to an η-random sampling oracle. In this case, if we draw samples (using an η-random sampling oracle) until we obtain k distinct elements of S then probability of an element getting selected is between $k/(1+\eta)|S|$ and $(1+\eta)k/|S|$. Thus if we draw a number k according to the Binomial distribution $B(|S|, p)$ and then draw k distinct elements at random from S using an η-random sampling oracle, then the probability that an element in S is selected is between $\sum_k \frac{k}{(1+\eta)|S|} \Pr[k \sim B(|S|, p)]$ and $\sum_k \frac{k(1+\eta)}{|S|} \Pr[k \sim B(|S|, p)]$, that is between $p/(1+\eta)$ and $p(1+\eta)$. Now, if we only have an (α, γ)-approximation of $|S|$ (instead of the exact value of $|S|$), it is still possible to design a sampling process where each item of S is selected independently with a probability that is between $p/2(1+\eta)$ and $p(1+\eta)(1+\alpha)^2$, which will be sufficient for our purposes. We detail this process in the next claim.

Claim 5.2. *Let S be any set and Z be an (α, γ)-approximation of $|S|$. For any $p \leq \frac{1}{2(1+\alpha)^2}$, consider the process: first draw a number k according to the Binomial distribution $\mathrm{Bin}(Z(1+\alpha), p)$ and then draw k distinct samples using an η-random sampling oracle from S. Then with probability at least $(1-\gamma)$ each element of S is picked independently and for any element $x \in S$*

$$\frac{p}{2(1+\eta)} \leq \Pr[x \textit{ is picked}] \leq (1+\alpha)^2 p(1+\eta), \tag{4}$$

assuming $S \geq 3\log 2(1+\eta)/p$

Claim 5.2 is similar to that of Claim 2.5. The proof of Claim 5.2 is presented in the Appendix.

We will need one more claim to prove the algorithm's correctness that estimates the size of Approximate-Delphic Sets. The claim follows from a standard application of Chernoff's bound.

Claim 5.3. *Let R be a set of N elements and each element of R is selected independently with some probability that is guaranteed to be between $\beta_1 p$ and $\beta_2 p$. Let P be the random variable that counts the number of selected items. Then, assuming $\beta_1 \leq 1 \leq \beta_2$,*

$$\Pr\left[(1-\varepsilon)\beta_1 pN \leq P \leq (1+\varepsilon)\beta_2 pN\right] \geq 1 - 2e^{-\varepsilon^2 pN\beta_1}.$$

Using Observation 5.1, Claim 5.2 and Claim 5.3 we now present the generalization of Vatic to handle Approximate-Delphic sets.

The algorithm to estimate the size of the union of the sets from an Approximate-Delphic family with access to a (α, γ, η)-Approximate-Delphic Oracle is presented in Ext-Vatic. The correctness and the space and the update time complexities of Ext-Vatic is presented in the following theorem which is restated.

Theorem 1.5. *There is a streaming algorithm, which we call Ext-Vatic that, given real numbers $\varepsilon, \delta < 1$, and a stream $\mathcal{S} = \langle S_1, S_2, \cdots, S_M \rangle$ of unknown length M where each $S_i \subseteq \Omega$ belongs to an Approximate-Delphic family, and access to an (α, γ, η)-Approximate-Delphic oracle for some α, γ, δ for members of the family, outputs a number in the range $[\frac{(1-\varepsilon)}{2(1+\eta)(1+\alpha)}|\bigcup_{i=1}^M S_i|, (1+\varepsilon)(1+\eta)(1+\alpha)|\bigcup_{i=1}^M S_i|]$ The worst case space complexity of the algorithm is $O((\log^3|\Omega|)\log(1/\delta) \cdot \frac{(1+\eta)}{\epsilon^2})$. The algorithm, while processing any item of the stream, makes*

$$\tilde{O}((\log^3|\Omega|)\log(1/\delta)\log(\frac{1}{1-\gamma})\frac{(1+\eta)}{\epsilon^2})$$

calls to the (α, γ, η)-Approximate-Delphic Oracle in the worst case.

Algorithm 2 Ext-Vatic

```
Initialize L = (log(8/δ)/ε²) · 2(1 + η)
Initialize B ← (L log(2|Ω|/δ))
Initialize Thresh₁ ← 3 log(2(1 + η)|Ω|/L)
Initialize Thresh₂ ← (1 + η) · Thresh₁ · log(8|Ω|/δ · Thresh₁)
Initialize X ← ∅
for i = 1 to M do
    for all (s, ∗) ∈ X do
        if s ∈ S_i then
            remove (s, ∗) from X
    for k = 1 to Thresh₂ do
        Pick a random sample y from S (using the η-sampling oracle)
        if y is not in Y then
            Y = Y ∪ {y}
    if |Y| ≤ Thresh₁ then
        E_i = |Y|
    else
        E_i = (1 + α)T_i; [T_i is an (α, (δ/4)/|Ω|)-approximation of |S_i|]
    Reset Y to ∅
    Set p ← 1/2(1 + α)²
    Pick N_i from the binomial distribution Bin(E_i(1 + α), p)
    while p > 1/2^⌈(|X|+N_i)/B⌉ and p ≥ L/|Ω| do
        N_i ← Bin(N_i, 1/2) and p ← p/2
    if p > L/|Ω| then
        Set K_i ← 4N_i · log(4Ω/δ)
        for k = 1 to K_i do
            y ← Sample(S_i)
            if |L| < N_i then
                L ← L ∪ {(y, p)}
        X ← X ∪ L;
Let p_0 = min{p_s | ∃s, (s, p_s) ∈ X}
for (s, p_s) ∈ X do
    With probability (1 − p_0/p_s) remove (s, p_s) from X
Output |X| / p(1+α)
```

6 APPLICATIONS

So far, we have presented key technical results in the context of Delphic and Approximate-Delphic sets in their generality and presented algorithms VATIC and EXT-VATIC. We also demonstrated that the streaming version of the well-known Klee's Measure Problem fits in the Delphic family framework (this has already been done in [33]). In this section, we discuss how algorithms VATIC and EXT-VATIC can be applied to a wide range of significant computational problems.

6.1 Applications of the Delphic Family Framework

We now briefly discuss streaming problems that fit the Delphic family framework. The descriptions of these problems, except that of the Hypervolume estimation problem, are based on [33], where the significance of these problems is discussed in some detail.

Hypervolume indicator estimation: Hypervolume indicator estimation is a special case of KMP wherein every rectangle has the origin $(0, 0, \dots 0)$ as a vertex. We define it as follows: A d-dimensional axis aligned rectangle $\mathbf{r}$ over an universe $\Omega = \Delta^d$, rooted at the origin, is defined as the set $[0, b_1] \times [0, b_2] \times \dots \times [0, b_d]$. Given a rectangle $\mathbf{r}$ rooted at origin, let $\mathsf{Range}(\mathbf{r})$ denote set of tuples $\{(x_1, \dots, x_d)\}$ where $0 \leq x_i \leq b_i$ and $x_i \in \Delta$. Such a d-dimensional rectangle can be succinctly represented by the tuple $(b_1, b_2, \cdots, b_d)$. Hypervolume indicator estimation problem is the following: Given a stream $\mathcal{R}$ of size M such that $\mathcal{R} = \langle \mathbf{r}_1, \mathbf{r}_2, \cdots \mathbf{r}_M \rangle$, where each item $\mathbf{r}_i$ is a d-dimensional rectangle rooted at the origin over $\Omega = \Delta^d$, give a (ε, δ)-approximation of the $\mathsf{Volume}\mathcal{R}$, the volume of $\mathcal{R}$.

Hypervolume indicator is employed to measure the quality of Pareto sets in the context of multi-objective optimization [35]. We point the readers to a recent survey [18] for details on this important quality measure and computational problems and algorithms related to it.

Test Coverage Estimation: For an n-bit string $\mathbf{a} = a_1 a_2 \cdots a_n \in \{0, 1\}^n$, the *t-coverage* of $\mathbf{a}$, denoted by $\mathsf{Cov}_t(\mathbf{a})$, is defined as

$$\mathsf{Cov}_t(\mathbf{a}) = \big\{(T, \mathbf{y}) \ \mid \ T \subset [n], |T| = t, \mathbf{y} \in \{0,1\}^t \text{ and the restriction of } \mathbf{a} \text{ to indices in } T \text{ gives } \mathbf{y}\}$$

The input is a stream $\mathcal{A}$ of size M such that $\mathcal{A} = \langle \mathbf{a}_1, \dots, \mathbf{a}_M \rangle$ where $\mathbf{a}_i \in \{0, 1\}^n$, the *t-coverage* of $\mathcal{A}$, denoted by $\mathsf{Cov}_t(\mathcal{A})$, is defined as $\mathsf{Cov}_t(\mathcal{A}) = \cup_{1 \leq i \leq M} \mathsf{Cov}_t(\mathbf{a}_i)$.

The test coverage estimation problem is: *Given a stream* $\mathcal{A} = \boldsymbol{a}_1, \cdots, \boldsymbol{a}_M$*, compute an* (ε, δ)*-approximation of* $|\mathsf{Cov}_t(\mathcal{A})|$ *for any given* t.

Observe that corresponding to every $\mathbf{a}_i$, we can construct the set $S_i = \mathsf{Cov}_t(\mathbf{a})$, which satisfies the desired properties of Delphic sets.

Model Counting for DNF: Let X be a set of n Boolean variables. A literal is a variable or its negation. A formula φ over X is in DNF if it is represented as a disjunction of conjunctions of literals. Each such conjunction is called a term, therefore, φ over M terms is represented as $T_1 \vee T_2 \vee \dots \vee T_M$. Let $\mathsf{Sol}(\varphi)$ represent the set of satisfying assignments of φ. The streaming version of the DNF model counting problem is the following: Given a DNF formula $\varphi = T_1 \vee T_2 \vee \dots \vee T_M$, as a stream $\langle T_1, \dots, T_M \rangle$ of M terms, compute an (ε, δ)-approximation of $|\mathsf{Sol}(\varphi)|$.

Corresponding to every term T_i, we can construct the set $S_i = \mathsf{Sol}(T_i)$, which satisfies the desired properties of Delphic sets.

6.2 Applications of the Approximate-Delphic Family Framwork

We now discuss natural problems that can be framed as set union estimation problems over the Approximate-Delphic family. In general, these problems are related to well-known computational problems for which exact counting is #P-hard, but there are efficient approximate counting algorithms. We briefly discuss some of them here without details about parameters.

Discrete volume of convex bodies: The problem is to compute a (ε, δ)-approximation of *discrete volume* (number of lattice points) of the union of a set of convex bodies in a set stream. An item in the stream is a list of vertices or facets of a polytope $\mathcal{P}$. Membership checking (i.e., to check whether $x \in \mathcal{P}$, i.e., whether x lies inside the polytope $\mathcal{P}$) can be accomplished in polynomial time. But, in its generality, even approximating the number of integer points in an arbitrary polytope is NP-hard. However, there are efficient sampling and approximate counting algorithms for special cases. An interesting and somewhat general case is when each polytope $\mathcal{P}$ is large: in particular, $\mathcal{P}$ is large enough to contain a ball of radius $\Omega(n\sqrt{\log m})$ where n is the dimension, and m is the number of facets. In this case, Kannan and Vempala gave polynomial-time algorithms for approximate uniform sampling and also to approximately count the number of lattice points of $\mathcal{P}$ within a constant factor [22].

Knapsack counting problem: #KNAP is the following problem: Given a non-negative vector $\mathbf{a} = (a_1, \dots, a_n)$ and non-negative integer b; count the number of $x \in \{0, 1\}^n$ so that $\sum_i a_i x_i \leq b$. In the set streaming problem, each item is a #KNAP instance and goal is to approximate the size of the union of the sets described by each instance. It is known that the exact counting is #P-hard. A good body of research has gone into designing approximate counting (and sampling) algorithms for #KNAP [11, 12, 16, 26]. In particular, [16] designed a deterministic fully polynomial time approximation scheme for the #KNAP and an algorithm to uniformly sample from the set described by an instance.

Boolean Circuits: As mentioned in Remark 1.6, Boolean circuits are general enough to be able to represent a large class of sets. In the set streaming setting, each item in the stream is a Boolean circuit C over n-bit binary strings. The problem is to give an (ε, δ)-approximation of the union of sets represented by all the circuits in the stream. While the problem of computing the exact size of the set represented by a Boolean circuit is #P-hard, the (α, γ, η)-Approximate-Delphic oracle can be implemented with $\mathsf{poly}(|C|, \log 1/\gamma, 1/\alpha, 1/\eta)$ calls to an NP oracle [20, 30].

7 CONCLUSION

In this paper, we present the first streaming algorithm for obtaining an (ε, δ)-approximation of the size of the union of Delphic sets using only $\mathsf{poly}(\log |\Omega|, \varepsilon^{-1}, \log \delta^{-1})$ worst-case space and update time complexity, independent of the stream size. We also extend our result to handle Approximate-Delphic sets. These two results answer two of the open problems from [33]. We would like to note that both our algorithms can be adapted to obtain approximate-uniform sampling algorithms from the union of the sets. While we achieved the broad goal of designing algorithms with no dependence on the stream size M for a large class of problems, there are more questions that need to be explored. A natural direction to explore would be to improve the space and update time complexity, in particular their dependence on $\log(|\Omega|)$. For special cases of Delphic sets such as DNF [32] and Distinct Elements [21], algorithms with only linear dependence on $\log(|\Omega|)$ for space complexity with $\mathrm{poly}(\log(|\Omega|))$ update time complexity are known (ignoring the dependence on ε and δ). It is worth remarking that there is lower bound of $\Omega((\log|\Omega| + \frac{1}{\varepsilon}) \cdot \log(1/\delta))$ for Distinct Elements. Trivially, this lower bound also holds for estimating the union of Delphic Sets. Bridging the gap between lower and upper bounds in the context of Delphic sets remains an important open question.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers of PODS 22 for valuable comments and suggestions that help improve the presentation of the paper. This work was supported in part by National Research Foundation Singapore under its NRF Fellowship Programme[NRF-NRFFAI1-2019-0004], Ministry of Education Singapore Tier 2 grant MOE-T2EP20121-0011, NSF CCF-2130608 and HDR:TRIPODS-1934884 awards.

REFERENCES

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

[2] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. 2002. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*. Springer, 1–10.

[3] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 2002. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proc. of SODA*. ACM/SIAM, 623–632.

[4] Jon Louis Bentley. 1977. *Algorithms for Klee's rectangle problems*. Technical Report. Technical Report, Computer.

[5] Karl Bringmann and Tobias Friedrich. 2010. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.* 43, 6-7 (2010), 601–610.

[6] Timothy M Chan. 2010. A (slightly) faster algorithm for Klee's measure problem. *Computational Geometry* 43, 3 (2010), 243–250.

[7] Eric Y Chen and Timothy M Chan. 2005. Space-efficient algorithms for Klee's measure problem. *algorithms* 3, 5 (2005), 6.

[8] Bogdan S Chlebus. 1998. On the Klee's measure problem in small dimensions. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 304–311.

[9] S. A. Cook. 1971. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing (STOC)*. ACM, New York, NY, USA, 151–158. https://doi.org/10.1145/800157.805047

[10] P. Dagum, R. Karp, M. Luby, and S. Ross. 2000. An optimal algorithm for Monte Carlo estimation. *SIAM Journal on computing* 29, 5 (2000), 1484–1496.

[11] Martin Dyer. 2003. Approximate Counting by Dynamic Programming. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing (STOC '03)*. Association for Computing Machinery, New York, NY, USA, 693–699. https://doi.org/10.1145/780542.780643

[12] Martin Dyer, Alan Frieze, Ravi Kannan, Ajai Kapoor, Ljubomir Perkovic, and Umesh Vazirani. 1993. A Mildly Exponential Time Algorithm for Approximating the Number of Solutions to a Multidimensional Knapsack Problem. *Combinatorics, Probability and Computing* 2, 3 (1993), 271–284. https://doi.org/10.1017/S0963548300000675

[13] Philippe Flajolet and G Nigel Martin. 1985. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences* 31, 2 (1985), 182–209.

[14] Michael L Fredman and Bruce Weide. 1978. On the complexity of computing the measure of $\bigcup$[ai, bi]. *Commun. ACM* 21, 7 (1978), 540–544.

[15] 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*. 281–291.

[16] Parikshit Gopalan, Adam Klivans, Raghu Meka, Daniel Štefankovic, Santosh Vempala, and Eric Vigoda. 2011. An FPTAS for #Knapsack and Related Counting Problems. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. 817–826. https://doi.org/10.1109/FOCS.2011.32

[17] Joachim Gudmundsson and Rasmus Pagh. 2017. Range-Efficient Consistent Sampling and Locality-Sensitive Hashing for Polygons. In *28th International Symposium on Algorithms and Computation, ISAAC 2017, December 9-12, 2017, Phuket, Thailand (LIPIcs)*, Yoshio Okamoto and Takeshi Tokuyama (Eds.), Vol. 92. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 42:1–42:13.

[18] Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. 2021. The Hypervolume Indicator: Computational Problems and Algorithms. *ACM Comput. Surv.* 54, 6, Article 119 (jul 2021), 42 pages. https://doi.org/10.1145/3453474

[19] J. Hartmanis and R.E. Stearns. 1966. *Algebraic Structure Theory of Sequential Machines*. Prentice Hall.

[20] M.R. Jerrum, L.G. Valiant, and V.V. Vazirani. 1986. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science* 43, 2-3 (1986), 169–188.

[21] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. 2010. An optimal algorithm for the distinct elements problem. In *Proc. of PODS*. ACM, 41–52.

[22] Ravi Kannan and Santosh Vempala. 1997. Sampling Lattice Points. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC '97)*. Association for Computing Machinery, New York, NY, USA, 696–700. https://doi.org/10.1145/258533.258665

[23] R.M. Karp and M. Luby. 1983. Monte-Carlo algorithms for enumeration and reliability problems. *Proc. of FOCS* (1983).

[24] Richard M Karp, Michael Luby, and Neal Madras. 1989. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms* 10, 3 (1989), 429 – 448. https://doi.org/10.1016/0196-6774(89)90038-2

[25] Victor Klee. 1977. Can the Measure of be Computed in Less than O (n log n) Steps? *The American Mathematical Monthly* 84, 4 (1977), 284–285.

[26] Ben Morris and Alistair Sinclair. 2004. Random Walks on Truncated Cubes and Sampling 0-1 Knapsack Solutions. *SIAM J. Comput.* 34, 1 (2004), 195–226. https://doi.org/10.1137/S0097539702411915 arXiv:https://doi.org/10.1137/S0097539702411915

[27] P. Motwani, R.and Raghavan. 1995. *Randomized algorithms*. Cambridge University Press, New York, NY, USA.

[28] Mark H Overmars and Chee-Keng Yap. 1991. New upper bounds in Klee's measure problem. *SIAM J. Comput.* 20, 6 (1991), 1034–1045.

[29] A. Pavan and Srikanta Tirthapura. 2007. Range-Efficient Counting of Distinct Elements in a Massive Data Stream. *SIAM J. Comput.* 37, 2 (2007), 359–379.

[30] L.J. Stockmeyer. 1974. *The complexity of decision procedures in Automata Theory and Logic*. Ph.D. Dissertation. MIT. Project MAC Technical Report TR-133.

[31] He Sun and Chung Keung Poon. 2007. Two Improved Range-Efficient Algorithms for F 0 Estimation. In *International Conference on Theory and Applications of Models of Computation*. Springer, 659–669.

[32] A. Pavan ⓡ N. V. Vinodchandran ⓡ Arnab Bhattacharyya ⓡ Kuldeep S. Meel. 2021. Model Counting meets F_0 Estimation. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*.

[33] Kuldeep S. Meel ⓡ N.V. Vinodchandran ⓡ Sourav Chakraborty. 2021. Estimating the Size of Unions of Sets in Streaming Models. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*.

[34] Srikanta Tirthapura and David P. Woodruff. 2012. Rectangle-efficient aggregation in spatial data streams. In *Proc. of PODS*. ACM, 283–294.

[35] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. 2007. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *International Conference on Evolutionary Multi-Criterion Optimization*. Springer, 862–876.

A PROOF OF CLAIM 2.5

Claim 2.5. *The sampling process $\mathcal{P}$ samples each element of S independently with probability p.*

Proof. For any $x \in S$, the probability of choosing x is $\sum_k \frac{k}{|S|} \Pr[k \sim B(|S|, p)]$. Using the definition of Binomial distribution we have

$$\begin{aligned}
\sum_{k=0}^{|S|} \frac{k}{|S|} \Pr\left[k \sim B(|S|, p)\right] &= \sum_{k=0}^{|S|} \frac{k}{|S|} \binom{|S|}{k} p^k (1-p)^{|S|-k} \\
&= \sum_{k=1}^{|S|} \binom{|S|-1}{k-1} p^k (1-p)^{|S|-k} \\
&= p \cdot \sum_{k=0}^{|S|-1} \binom{|S|-1}{k} p^k (1-p)^{|S|-1-k} \\
&= p
\end{aligned}$$

Now, to prove that each element of S is chosen independently let us calculate the probability of choosing any specific $x_1, \ldots, x_t$ from S. Note that, when one picks a subset of size k from S, probability that all $x_1, \ldots, x_t$ is picked is 0 if $k < t$ and is $\binom{k}{t}/\binom{|S|}{t}$ otherwise. So, the probability that our process will of choose $x_1, \ldots, x_t$ is

$$\begin{aligned}
&\sum_{k=0}^{|S|} \frac{\binom{k}{t}}{\binom{|S|}{t}} \Pr\left[k \sim B(|S|, p)\right] \\
= &\sum_{k=t}^{|S|} \frac{\binom{k}{t}}{\binom{|S|}{t}} \Pr\left[k \sim B(|S|, p)\right] \\
= &\sum_{k=t}^{|S|} \frac{\binom{k}{t}}{\binom{|S|}{t}} k p^k (1-p)^{|S|-k} \\
= &\sum_{k=t}^{|S|} \binom{|S|-t}{k-t} p^k (1-p)^{|S|-k} \\
= &\; p^t \cdot \sum_{k=0}^{|S|-t} \binom{|S|-t}{k} p^k (1-p)^{|S|-t-k} \\
= &\; p^t
\end{aligned}$$

Thus, for any set of t elements in S probability that the t elements are chosen is p^t. This proves that all the items of S are chosen independently with probability p.

□

B PROOF OF CLAIM 5.2

Claim 5.2. *Let S be any set and Z be an (α, γ)-approximation of $|S|$. For any $p \le \frac{1}{2(1+\alpha)^2}$, consider the process: first draw a number k according to the Binomial distribution* $\mathrm{Bin}(Z(1+\alpha), p)$ *and then draw k distinct samples using an η-random sampling oracle from S. Then with probability at least $(1-\gamma)$ each element of S is picked independently and for any element $x \in S$*

$$\frac{p}{2(1+\eta)} \le \Pr[x \textit{ is picked}] \le (1+\alpha)^2 p (1+\eta), \tag{4}$$

assuming $S \ge 3 \log 2(1+\eta)/p$

Proof. Since Z is an (α, γ)-approximation of $|S|$, by definition we have with probability at least $(1-\gamma)$, $|S|/(1+\alpha) \le Z \le (1+\alpha)|S|$. In the rest of the proof we will show that Equation 4 holds assuming, $|S|/(1+\alpha) \le Z \le (1+\alpha)|S|$. The Claim will thus follow. We now, prove the upper and lower bound on $\Pr[x \text{ is picked}]$ in the Equation 4.

Upper bound: For any $x \in S$, the probability of x getting selected is $\le \sum_k \frac{k(1+\eta)}{|S|} \Pr[k \sim B(Z(1+\alpha), p)]$ which is less than or equal to $p\frac{Z(1+\alpha)}{|S|}(1+\eta)$ (by identical argument as in the proof of Claim 2.5). Since $Z \le (1+\alpha)|S|$ the above quantity is less than $(1+\alpha)^2 p(1+\eta)$ with probability $\ge (1-\gamma)$.

Lower bound: On the other hand, if it so happens that the number k drawn from $\mathrm{Bin}(Z(1+\alpha), p)$ is bigger than the actual size of the set S then drawing k distinct elements from S would be impossible. But since $p \le \frac{1}{2(1+\alpha)^2}$ and $S \ge 3 \log 2(1+\eta)/p$ then by Chernoff bound we have that $\Pr[k > |S|] < p/2(1+\eta)$. Thus the probability that an element x is drawn is

$$\begin{aligned}
&\ge \sum_{k=0}^{|S|} \frac{k}{|S|(1+\eta)} \Pr[k \sim B(Z(1+\alpha), p)] \\
&\ge \sum_{k=0}^{Z(1+\alpha)} \frac{k}{|S|(1+\eta)} \Pr[k \sim B(Z(1+\alpha), p)] - \frac{p}{2(1+\eta)} \\
&= \frac{p}{2(1+\eta)}
\end{aligned}$$

The final equality follows from identical argument as in the proof of Claim 2.5. The proof that the elements of S are picked independently is follows from identical argument as in the proof of Claim 2.5.

□

C PROOF OF COUPON COLLECTOR PROBLEM

Theorem C.1 (Coupon Collector Problem). *Given access to uniform random samples from a set T and a number $r \le |T|$, let Z_r be a random variable that stands for the number of independent uniform random samples from T needed before we get r distinct samples from T. Then for any $\beta \ge 1$*

$$\Pr\left[Z_r > \beta r \log r\right] \le r^{-(\beta/2)+1}.$$

Proof. Let us divide the elements in T into $(r+1$ number of disjoint buckets $B_0, B_1, \ldots, B_r$ of size $[|T|/r]$, where for all $i \ne 0$ the size of the bucket B_i is $[|T|/r]$ and the B_0 contains the rest of the items, that is $\{|T|/r\}$ items. Here we denote by $[x]$ the largest integer less than or equal to x and $\{x\}$ denots $x - [x]$. Let $\{|T|/r\}$ be t and $[|T|/r]$ be s. . Note $0 \le t < r$ and $|T| = sr + t$, and hence $sr \le |T|/2$.

Let us draw a set of $\beta r \log r$ independent samples from the set T. Note that this means that with probability $s/|T|$ an element from a bucket B_i is drawm. Let A_i denote the random variable indicating whether an element from bucket B_i is not drawn. Note that

$$\Pr[A_i] = \left(1 - \frac{s}{|T|}\right)^{\beta r \log r} = \left(\frac{1}{e}\right)^{\beta \frac{sr}{|T|} \log r} \le r^{-\frac{sr}{|T|}\beta}.$$

So the probability that the random variable Z_r is more than $\beta r \log r$ is less than the probability that some element of each of the

Algorithm 3 Ext-APS-Estimator

1: Initialize $\mathsf{Thresh}_1 \leftarrow \left(\frac{\log(8/\delta)+\log M}{\varepsilon^2}\right)$
2: Initialize $\mathsf{Thresh}_2 \leftarrow 3\log(2|\Omega|(1+\eta))$
3: Initialize $\mathsf{Thresh}_3 \leftarrow (1+\eta)\cdot \mathsf{Thresh}_2 \cdot \log(|\Omega|\cdot \mathsf{Thresh}_2)$
4: Initialize $p \leftarrow 1/2(1+\alpha)^2$
5: Initialize $\mathcal{X}, \mathcal{Y} \leftarrow \emptyset$
6: **for** $i = 1$ to M **do**
7:     **for** all $(s,*) \in \mathcal{X}$ **do**
8:         **if** $s \in S_i$ **then**
9:             remove $(s,*)$ from $\mathcal{X}$
10:     **for** k = 1 to Thresh_3 **do**
11:         Pick a random sample y from S (using the η-sampler)
12:         **if** y is not in $\mathcal{Y}$ **then**
13:             $\mathcal{Y} = \mathcal{Y} \cup \{y\}$
14:     **if** $|\mathcal{Y}| \le \mathsf{Thresh}_2$ **then**
15:         $E_i = |\mathcal{Y}|$
16:     **else** $E_i = (1+\alpha)T_i$; [T_i is an $(\alpha, \frac{\delta/2}{|\Omega|})$-approximation of $|S_i|$]
17:     Reset $\mathcal{Y}$ to $\emptyset$
18:     Pick a number N_i from the binomial distribution $B(E_i, p)$
19:     **while** $N_i + |\mathcal{X}|$ is more than Thresh_1 **do**
20:         Throw away each element of $\mathcal{X}$ with probability $1/2$
21:         $N_i = B(N_i, 1/2)$ and $p = p/2$
22:     **for** k = 1 to N_i **do**
23:         Draw a random sample y from S_i such that $x \notin \mathcal{X}$
24:         Add x to $\mathcal{X}$.
25: Output $\frac{|\mathcal{X}|}{p(1+\alpha)}$

buckets $B_1, \ldots, B_r$ is not drawn when $\beta r \log r$ elements are drawn uniformly and independently at random. Thus,

$$\Pr\left[Z_r > \beta r \log r\right] \le \Pr[\cup_{i=1}^r A_i] \le \sum_{i=1}^r \Pr[A_i] = r^{-\frac{sr}{|T|}|\beta+1}.$$

Since $sr \ge |T|/2$ so have $\Pr\left[Z_r > \beta r \log r\right] \le r^{-(\beta/2)+1}$.

□

D EXTENSION OF THE APS-ESTIMATOR ALGORITHM (FROM [33]) TO APPROXIMATE-DELPHIC SETS

The technique used in the proof of Theorem 1.5 can be used to extend the algorithm APS-Estimator (from [33]) to work with (α,γ,η)-Approximate-Delphic sets. Ext-APS-Estimator is the extended algorithm. It also uses a slightly different implementation of the algorithm as compared to that in [33]. The proof the following theorem follows using exactly the same arguments as used in Theorem 1.5, and thus we skip the proof of this theorem. Note that, as in [33], the algorithm Ext-APS-Estimator needs to know the size of the stream in advance and the complexity depends on the size of the stream.

Theorem D.1. *Given any reals numbers $0 < \varepsilon, \delta < 1$, and a stream $\mathcal{S} = \langle S_1, S_2 \cdots, S_M\rangle$ wherein each $S_i \subseteq \Omega$ belongs to an Approximate-Delphic family, the algorithm* Ext-APS-Estimator, *given access to an (α,γ,η)-Approximate-Delphic Oracle, outputs a number that is between $\frac{(1-\varepsilon)}{2(1+\eta)(1+\alpha)}|\bigcup_{i=1}^M S_i|$ and $(1+\varepsilon)(1+\eta)(1+\alpha)|\bigcup_{i=1}^M S_i|$. The algorithm has worst case space complexity $O\left(\log\left(\frac{|\Omega|}{\delta}\right)\cdot\frac{(1+\eta)}{\epsilon^2}\right)$. For the update time the number of calls to the (α,γ,η)-Approximate-Delphic Oracle is $\tilde{O}\left((1+\eta)\log^2(|\Omega|)\cdot\log(1/\delta\gamma)\right)$.*

E PROOF OF THEOREM 1.5

Proof. We first prove the correctness of the algorithm. Note that the algorithm is exactly same as Vatic except following few points:

- The constants Thresh_1 and Thresh_2 are so set such that from Observation 5.1 we have: after the **for** loop in Line 10-13 is completed, while processing the set S_i, the number of elements in $\mathcal{Y}$ is at least $\min\{|S_i|, \mathsf{Thresh}_1\}$ with probability $\ge (1-\delta/8|\Omega|)$.
- Thus after the **if**-**else** condition in Line 14-17 $E_i = |S_i|$ if $|S_i| \le \mathsf{Thresh}_1$ and else with probability at least $(1-\delta/8|\Omega|)$, $\frac{|S_i|}{(1+\alpha)} \le E_i \le |S_i|(1+\alpha)$.
- The constant Thresh_1 is so set that with $p \le 1/2(1+\alpha)^2$ using Claim 5.2 one can see that in Line 20- 29 each element in S_i is added to $\mathcal{X}$ independently with probability that is between $p/2(1+\eta)$ and $(1+\alpha)^2 p(1+\eta)$.

Now following the same argument as in proof of Theorem 1.2 we see that at the end of the stream for any element of $x \in \cup_i S_i$ is (x, p_x) is in the set $\mathcal{X}$ with probability between $p_x/2(1+\eta)$ and $(1+\alpha)^2 p_x(1+\eta)$ and $p_x \ge L/|\cup_i S_i|$. Thus from Claim 5.3 we have that with probability $\ge (1-\frac{\delta}{4}$

$$\frac{(1-\varepsilon)}{2(1+\eta)}|\cup_i S_i| \le \frac{|\mathcal{X}|}{p} \le (1+\alpha)^2(1+\eta)|\cup_i S_i|.$$

By using union bound over all the possible errors we bound the total error probability to $\le \delta$.

The space complexity is obvious from the pseudocode. The update time complexity also follows easily. The only thing to keep in mind is that in Line 17 an access to an $\left(\alpha, \frac{\delta/4}{|\Omega|}, \eta\right)$-Approximate-Delphic oracle is needed and this, as observed in Observation 5.1, needs $\log(4|\Omega|/\delta)$ calls to an (α,γ,η)-Approximate-Delphic Oracle.

□

F PROOF OF CASCADE BINOMIAL SAMPLING

Our sampling process involves sampling the binomial distribution $\mathrm{Bin}(n,p)$ for a positive integer n (cardinality of a set in the stream) and a probability p that is adaptively chosen. In general to sample the distribution $\mathrm{Bin}(n,pq)$ the process we employ a cascading process: first sample $\mathrm{Bin}(n,p)$ to get a number l and then sample $\mathrm{Bin}(l,q)$. Let $\mathcal{S}$ denote this process. For completeness we give proof of correctness that $\mathcal{S}$ is same as sampling from $\mathrm{Bin}(n,pq)$.

Theorem F.1. *Let n be a positive integer and $0 \le p, q \le 1$ be probability values. Consider the following sampling process $\mathcal{S}$: First get l according to $\mathrm{Bin}(n,p)$ and then get k according to $\mathrm{Bin}(l,q)$. Then the sampling process $\mathcal{S}$ is same as sampling $\mathrm{Bin}(n,pq)$.*

Proof. We will show that $\Pr(k \leftarrow \mathcal{S}) = \Pr(k \leftarrow \mathrm{Bin}(n,pq)) = \binom{n}{k}(pq)^k(1-pq)^{n-k}$.

□

$$
\begin{aligned}
\Pr(k \leftarrow \mathcal{S}) &= \sum_{l=0}^{n} \Pr\left(k \leftarrow \mathrm{Bin}(l,q) \mid l \leftarrow \mathrm{Bin}(n,p)\right) \cdot \Pr(l \leftarrow \mathrm{Bin}(n,p)) \\
&= \sum_{l=0}^{n} \binom{l}{k} q^k (1-q)^{l-k} \cdot \binom{n}{l} p^l (1-p)^{n-l} \\
&= \sum_{l=0}^{n} \binom{n}{l}\binom{l}{k} p^l (1-p)^{n-l} q^k (1-q)^{l-k} \\
&= \sum_{l=0}^{n} \binom{n}{k}\binom{n-k}{l-k} p^l (1-p)^{n-l} q^k (1-q)^{l-k} \\
&= \binom{n}{k} \sum_{l \geq k}^{n} \binom{n-k}{l-k} p^l (1-p)^{n-l} q^k (1-q)^{l-k} \\
&= \binom{n}{k} \sum_{r=0}^{n-k} \binom{n-k}{r} p^{r+k} (1-p)^{n-r-k} q^k (1-q)^r \\
&= \binom{n}{k} (pq)^k \sum_{r=0}^{n-k} \binom{n-k}{r} p^r (1-p)^{n-r-k} (1-q)^r \\
&= \binom{n}{k} (pq)^k \sum_{r=0}^{n-k} \binom{n-k}{r} p^r (1-q)^r (1-p)^{n-r-k} \\
&= \binom{n}{k} (pq)^k \left((1-q)p + 1 - p\right)^{n-k} \\
&= \binom{n}{k} (pq)^k (1-pq)^{n-k} \\
&= \Pr\left(k \leftarrow \mathrm{Bin}(n,pq)\right)
\end{aligned}
$$