A Framework for Estimating Stream Expression Cardinalities

Anirban Dasgupta[1], Kevin J. Lang[2], Lee Rhodes[3], and Justin Thaler[4]

1 **IIT Gandhinagar, Gandhinagar, India**
anirban.dasgupta@gmail.com
2 **Yahoo Inc., 701 First Ave, Sunnyvale, CA, USA**
langk@yahoo-inc.com
3 **Yahoo Inc., 701 First Ave, Sunnyvale, CA, USA**
lrhodes@yahoo-inc.com
4 **Yahoo Inc., New York, NY, USA**
jthaler@fas.harvard.edu

Abstract

Given m distributed data streams $A_1, \ldots, A_m$, we consider the problem of estimating the number of unique identifiers in streams defined by set expressions over $A_1, \ldots, A_m$. We identify a broad class of algorithms for solving this problem, and show that the estimators output by any algorithm in this class are perfectly unbiased and satisfy strong variance bounds. Our analysis unifies and generalizes a variety of earlier results in the literature. To demonstrate its generality, we describe several novel sampling algorithms in our class, and show that they achieve a novel tradeoff between accuracy, space usage, update speed, and applicability.

1998 ACM Subject Classification G.3 [Probability and Statistics] Probabilistic algorithms, Stochastic processes, H.2.8 [Database Applications] Data mining

Keywords and phrases sketching, data stream algorithms, mergeability, distinct elements, set operations

Digital Object Identifier 10.4230/LIPIcs.ICDT.2016.6

1 Introduction

Consider an internet company that monitors the traffic flowing over its network by placing a sensor at each ingress and egress point. Because the volume of traffic is large, each sensor stores only a small *sample* of the observed traffic, using some simple sampling procedure. At some later point, the company decides that it wishes to estimate the number of unique users who satisfy a certain property P and have communicated over its network. We refer to this as the DistinctOnSubPopulation$_P$ problem, or Distinct$_P$ for short. How can the company combine the samples computed by each sensor, in order to accurately estimate the answer to this query?

In the case that P is the trivial property that is satisfied by all users, the answer to the query is simply the number of DistinctElements in the traffic stream, or Distinct for short. The problem of designing streaming algorithms and sampling procedures for estimating DistinctElements has been the subject of intense study. In general, however, P may be significantly more complicated than the trivial property, and may not be known until query time. For example, the company may want to estimate the number of (unique) men in a certain age range, from a specified country, who accessed a certain set of websites

during a designated time period, while excluding IP addresses belonging to a designated blacklist. This more general setting, where P is a nontrivial ad hoc property, has received somewhat less attention than the basic DISTINCT problem.

In this paper, our goal is to identify a simple method for combining the samples from each sensor, so that the following holds. As long as each sensor is using a sampling procedure that satisfies a certain mild technical condition, then for any property P, the combining procedure outputs an estimate for the DISTINCT$_P$ problem that is unbiased. Moreover, its variance should be bounded by that of the individual sensors' sampling procedures.[1]

For reasons that will become clear later, we refer to our proposed combining procedure as the *Theta-Sketch Framework*, and we refer to the mild technical condition that each sampling procedure must satisfy to guarantee unbiasedness as 1-*Goodness*. If the sampling procedures satisfy an additional property that we refer to as *monotonicity*, then the variance of the estimate output by the combining procedure is guaranteed to satisfy the desired variance bound. The Theta-Sketch Framework, and our analysis of it, unifies and generalizes a variety of results in the literature (see Section 2.5 for details).

The Importance of Generality. As we will see, there is a huge array of sampling procedures that the sensors could use. Each procedure comes with a unique tradeoff between accuracy, space requirements, update speed, and simplicity. Moreover, some of these procedures come with additional desirable properties, while others do not. We would like to support as many sampling procedures as possible, because the best one to use in any given given setting will depend on the relative importance of each resource in that setting.

Handling Set Expressions. The scenario described above can be modeled as follows. Each sensor observes a stream of identifiers A_j from a data universe of size n, and the goal is to estimate the number of distinct identifiers that satisfy property P in the combined stream $U = \cup_j A_j$. In full generality, we may wish to handle more complicated set expressions applied to the constituent streams, other than set-union. For example, we may have m streams of identifiers $A_1, \ldots, A_m$, and wish to estimate the number of distinct identifiers satisfying property P that appear in *all streams*. The Theta-Sketch Framework can be naturally extended to provide estimates for such queries. Our analysis applies to any sequence of set operations on the A_j's, but we restrict our attention to set-union and set-intersection throughout the paper for simplicity.

2 Preliminaries, Background, and Contributions

2.1 Notation and Assumptions

Streams and Set Operations. Throughout, A denotes a stream of identifiers from a data universe $[n] := \{1, \ldots, n\}$. We view any *property* P on identifiers as a subset of $[n]$, and let $n_{P,A} := \text{DISTINCT}_P(A)$ denote the number of distinct identifiers that appear in A and satisfy P. For brevity, we let n_A denote DISTINCT(A). When working in a multi-stream setting, $A_1, \ldots, A_m$ denote m streams of identifiers from $[n]$, $U := \cup_{j=1}^{m} A_j$ will denote the concatenation of the m input streams, while $I := \cap_{j=1}^{m} A_j$ denotes the set of identifiers that

[1] More precisely, we are interested in showing that the variance of the returned estimate is at most that of the (hypothetical) estimator obtained by running each individual sensor's sampling algorithm on the concatenated stream $A_1 \circ \cdots \circ A_m$. We refer to the latter estimator as "hypothetical" because it is typically infeasible to materialize the concatenated stream in distributed environments.

appear at least once in all m streams. Because we are interested only in *distinct* counts, it does not matter for definitional purposes whether we view U and I as sets, or as multisets. For any property $P\colon [n] \to \{0,1\}$, $n_{P,U} := \textsc{Distinct}_P(U)$ and $n_{P,I} := \textsc{Distinct}_P(I)$, while $n_U := \textsc{Distinct}(U)$ and $n_I := \textsc{Distinct}(I)$.

Hash Functions. For simplicity and clarity, and following prior work (e.g. [4, 5]), we assume throughout that the sketching and sampling algorithms make use of a perfectly random hash function h mapping the data universe $[n]$ to the open interval $(0,1)$. That is, for each $x \in [n]$, $h(x)$ is a uniform random number in $(0,1)$. Given a subset of hash values S computed from a stream A, and a property $P \subseteq [n]$, $P(S)$ denotes the subset of hash values in S whose corresponding identifiers in $[n]$ satisfy P. Finally, given a stream A, the notation X^{n_A} refers to the set of hash values obtained by mapping a hash function h over the n_A distinct identifiers in A.

2.2 Prior Art: Sketching Procedures for Distinct Queries

There is a sizeable literature on streaming algorithms for estimating the number of distinct elements in a single data stream. Some, but not all, of these algorithms can be modified to solve the $\textsc{Distinct}_P$ problem for general properties P. Depending on which functionality is required, systems based on HyperLogLog Sketches, K'th Minimum Value (KMV) Sketches, and Adaptive Sampling represent the state of the art for practical systems [11].[2] For clarity of exposition, and due to space constraints, we defer a more thorough overview of these algorithms to the full version of the paper [6]. Here, we briefly review the main concepts and relevant properties of each.

HLL: HyperLogLog Sketches. HLL is a sketching algorithm for the vanilla $\textsc{Distinct}$ problem. Its accuracy per bit is superior to the KMV and Adaptive Sampling algorithms described below. However, unlike KMV and Adaptive Sampling, it is not known how to extend the HLL sketch to estimate $n_{P,A}$ for general properties P (unless, of course, P is known prior to stream processing).

KMV: K'th Minimum Value Sketches. The KMV sketching procedure for estimating $\textsc{Distinct}(A)$ works as follows. While processing an input stream A, KMV keeps track of the set S of the k smallest unique hashed values of stream elements. The update time of a heap-based implementation of KMV is $O(\log k)$. The KMV estimator for $\textsc{Distinct}(A)$ is: $\text{KMV}_A = k/m_{k+1}$, where m_{k+1} denotes the $k{+}1^{\text{st}}$ smallest unique hash value.[3] It has been proved by [4], [10], and others, that $E(\text{KMV}_A) = n_A$, and $\sigma^2(\text{KMV}_A) = \frac{n_A^2 - k\, n_A}{k-1} < \frac{n_A^2}{k-1}$. Duffield et al. [7] proposed to change the heap-based implementation of the KMV sketching algorithm to an implementation based on quickselect [12]. This reduces the sketch update cost from $O(\log k)$ to amortized $O(1)$. However, this $O(1)$ hides a larger constant than competing methods. At the cost of storing the sampled identifiers, and not just their hash values, the KMV sketching procedure can be extended to estimate $n_{P,A}$ for any property $P \subseteq [n]$.

[2] Algorithms with better asymptotic bit-complexity are known [13], but they do not match the practical performance of the algorithms discussed here.

[3] Some works use the estimate k/m_k, *e.g.* [3]. We use k/m_{k+1} because it is unbiased, and for consistency with the work of Cohen and Kaplan [5] described below.

Adaptive Sampling. Adaptive Sampling maintains a sampling level $i \geq 0$, and the set S of all hash values less than 2^{-i}; whenever $|S|$ exceeds a pre-specified size limit, i is incremented and S is scanned discarding any hash value that is now too big. Because a simple scan is cheaper than running quickselect, an implementation of this scheme is typically faster than KMV. The estimator of n_A is $\text{Adapt}_A = |S|/2^{-i}$. It has been proved by [8] that this estimator is unbiased, and that $\sigma^2(\text{Adapt}_A) \approx 1.44(n_A^2/(k-1))$, where the approximation sign hides oscillations caused by the periodic culling of S. Like KMV, Adaptive Sampling can be extended to estimate $n_{P,A}$ for any property P. Although the stream processing speed of Adaptive Sampling is excellent, the fact that its accuracy oscillates as n_A increases is a shortcoming.

HLL for set operations on streams. HLL can be directly adapted to handle set-union. For set-intersection, the relevant adaptation uses the inclusion/exclusion principle. However, the variance of this estimate is approximately a factor of n_U/n_I worse than the variance achieved by the multiKMV algorithm described below. When $n_I \ll n_U$, this penalty factor overwhelms HLL's fundamentally good accuracy per bit.

KMV for set operations on streams. Given streams $A_1, \ldots, A_m$, let S_j denote the KMV sketch computed from stream A_j. A trivial way to use these sketches to estimate the number of distinct items n_U in the union stream U is to let M'_U denote the $(k+1)^{\text{st}}$ smallest value in the union of the sketches, and let $S'_U = \{x \in \cup_j S_j : x < M'_U\}$. Then S'_U is identical to the sketch that would have been obtained by running KMV directly on the concatenated stream $A_1 \circ \ldots, A_m$, and hence $\text{KMV}_{P,U} := k/M'_U$ is an unbiased estimator for n_U, by the same analysis as in the single-stream setting. We refer to this procedure as the "non-growing union rule."

Intuitively, the non-growing union rule does not use all of the information available to it. The sets S_j contain up to $k \cdot M$ distinct samples in total, but S'_U ignores all but the k smallest samples. With this in mind, Cohen and Kaplan [5] proposed the following adaptation of KMV to handle unions of multiple streams. We denote their algorithm by multiKMV, and also refer to it as the "growing union rule". Define $M_U = \min_{j=1}^m M_j$, and $S_U = \{x \in \cup_j S_j : x < M_U\}$. Then n_U is estimated by $\text{multiKMV}_U := |S_U|/M_U$, and $n_{P,U}$ is estimated by $\text{multiKMV}_{P,U} := |P(S_U)|/M_U$.

At first glance, it may seem obvious that the growing union rule yields an estimator that is "at least as good" as the non-growing union, since the growing union rule makes use of at least as many samples as the non-growing rule. However, it is by no means trivial to prove that $\text{multiKMV}_{P,U}$ is unbiased, nor that its variance is dominated by that of the non-growing union rule. Nonetheless, [5] managed to prove this: they showed that $\text{multiKMV}_{P,U}$ is unbiased and has variance that is dominated by the variance of $\text{KMV}_{P,U}$:

$$\sigma^2(\text{multiKMV}_{P,U}) \leq \sigma^2(\text{KMV}_{P,U}). \tag{1}$$

As observed in [5], multiKMV can be adapted in a similar manner to handle set-intersections (see Section 3.7 for details).

Adaptive Sampling for set operations on streams. Adaptive Sampling can handle set unions and intersections with a similar "growing union rule". Specifically, let $M_U := \min_{j=1}^m (2^{-i})_j$. Here, $(2^{-i})_j$ denotes the threshold for discarding hash values that was computed by the jth Adaptive Sampling sketch. We refer to this algorithm as multiAdapt. [9] proved epsilon-delta bounds on the error of $\text{multiAdapt}_{P,U}$, but did not derive expressions

for mean or variance. However, multiAdapt and multiKMV are both special cases of our Theta-Sketch Framework, and in Section 3 we will prove (apparently for the first time) that multiAdapt$_{P,U}$ is unbiased, and satisfies strong variance bounds. These results have the following two advantages over the epsilon-delta bounds of [9]. First, proving unbiasedness is crucial for obtaining estimators for distinct counts over subpopulations: these estimators are analyzed as a sum of a huge number of per-item estimates (see Theorem 11 for details), and biases add up. Second, variance bounds enable derivation of confidence intervals that an epsilon-delta guarantee cannot provide, unless the guarantee holds for many values of delta simultaneously.

2.3 Overview of the Theta-Sketch Framework

In this overview, we describe the Theta-Sketch Framework in the multi-stream setting where the goal is to output $n_{P,U}$, where $U = \cup_{j=1}^{m} A_j$ (we define the framework formally in Section 2.4). That is, the goal is to identify a very large class of sampling algorithms that can run on each constituent stream A_j, as well as a "universal" method for combining the samples from each A_j to obtain a good estimator for $n_{P,U}$. We clarify that the Theta-Sketch Framework, and our analysis of it, yields unbiased estimators that are interesting even in the single-stream case, where $m = 1$.

We begin by noting the striking similarities between the multiKMV and multiAdapt algorithms outlined in Section 2.2. In both cases, a sketch can be viewed as pair (θ, S) where θ is a certain threshold that depends on the stream, and S is a set of hash values which are all strictly less than θ. In this view, both schemes use the same estimator $|S|/\theta$, and also the same growing union rule for combining samples from multiple streams. The only difference lies in their respective rules for mapping streams to thresholds θ. The Theta-Sketch Framework formalizes this pattern of similarities and differences.

The assumed form of the single-stream sampling algorithms. The Theta-Sketch Framework demands that each constituent stream A_j be processed by a sampling algorithm samp$_j$ of the following form. While processing A_j, samp$_j$ evaluates a "threshold choosing function" (TCF) $T^{(j)}(A_j)$. The final state of samp$_j$ must be of the form $(\theta_j := T^{(j)}(A_j), S)$, where S is the set of all hash values strictly less than θ_j that were observed while processing A_j. If we want to estimate $n_{P,U}$ for non-trivial properties P, then samp$_j$ must also store the corresponding identifier that hashed to each value in S. Note that the framework itself does not specify the threshold-choosing functions $T^{(j)}$. Rather, any specification of the TCFs $T^{(j)}$ defines a particular instantiation of the framework.

▶ Remark. It might appear from Algorithm 1 that for any TCF $T^{(j)}$, the function samp$_j[T^{(j)}]$ makes two passes over the input stream: one to compute θ_j, and another to compute S_j. However, in all of the instantiations we consider, both operations can be performed in a single pass.

The universal combining rule. Given the states $(\theta_j := T^{(j)}(A_j), S_j)$ of each of the m sampling algorithms when run on the streams $A_1, \ldots, A_m$, define $\theta_U := \min_{j=1}^{m} \theta_j$, and $S_U := \{x \in \cup_j S_j : x < \theta_U\}$ (see the function ThetaUnion in Algorithm 1). Then n_U is estimated by $\hat{n}_U := |S_U|/\theta_U$, and $n_{P,U}$ as $\hat{n}_{P,U} := |P(S_U)|/\theta_U$ (see the function EstimateOnSubPopulation in Algorithm 1).

The analysis. Our analysis shows that, so long as each threshold-choosing function $T^{(j)}$ satisfies a mild technical condition that we call *1-Goodness*, then $\hat{n}_{P,U}$ is unbiased. We also

Algorithm 1 Theta Sketch Framework for estimating $n_{P,U}$. The framework is parameterized by choice of TCF's $T^{(j)}(k,A_j,h)$, one for each input stream.

1: **Definition:** Function $\text{samp}_j[T^{(j)}](k,\ A_j,\ h)$
  2: $\theta_j \leftarrow \mathrm{T}^{(j)}(k, A_j, h)$
  3: $S_j \leftarrow \{(x \in h(A_j)) < \theta_j\}$.
  4: **return** (θ_j, S_j).
5: **Definition:** Function ThetaUnion(Theta Sketches $\{(\theta_j, S_j)\}$)
  6: $\theta_U \leftarrow \min\{\theta_j\}$.
  7: $S_U \leftarrow \{(x \in (\cup S_j)) < \theta_U\}$.
  8: **return** (θ_U, S_U).
9: **Definition:** Function EstimateOnSubPopulation(Theta Sketch (θ, S) produced from stream A, Property P mapping identifiers to $\{0,1\}$)
  10: **return** $\hat{n}_{A,P} := \frac{|P(S)|}{\theta}$.

show that if each $T^{(j)}$ satisfies a certain additional condition that we call *monotonicity*, then $\hat{n}_{P,U}$ satisfies strong variance bounds (analogous to the bound of Equation (1) for KMV). Our analysis is arguably surprising, because 1-Goodness does not imply certain properties that have traditionally been considered important, such as permutation invariance, or S being a uniform random sample of the hashed unique items of the input stream.

Applicability. To demonstrate the generality of our analysis, we identify several valid instantiations of the Theta-Sketch Framework. First, we show that the TCF's used in KMV and Adaptive Sampling both satisfy 1-Goodness and monotonicity, implying that multiKMV and multiAdapt are both unbiased and satisfy the aforementioned variance bounds. For multiKMV, this is a reproof of Cohen and Kaplan's results [5], but for multiAdapt the results are new. Second, we identify a variant of KMV that we call pKMV, which is useful in multi-stream settings where the lengths of constituent streams are highly skewed. We show that pKMV satisfies both 1-Goodness and monotonicity. Third, we introduce a new sampling procedure that we call the *Alpha Algorithm*. Unlike earlier algorithms, the Alpha Algorithm's final state actually depends on the stream order, yet we show that it satisfies 1-Goodness, and hence is unbiased in both the single- and multi-stream settings. We also establish variance bounds on the Alpha Algorithm in the single-stream setting. We show experimentally that the Alpha Algorithm, in both the single- and multi-stream settings, achieves a novel tradeoff between accuracy, space usage, update speed, and applicability.

Unlike KMV and Adaptive Sampling, the Alpha Algorithm does not satisfy monotonicity in general. In fact, we have identified contrived examples in the multi-stream setting on which the aforementioned variance bounds are (weakly) violated. The Alpha Algorithm does, however, satisfy monotonicity under the promise that the $A_1, \ldots, A_m$ are pairwise disjoint, implying variance bounds in this case. Our experiments suggest that, in practice, the normalized variance in the multi-stream setting is not much larger than in the pairwise disjoint case.

Deployment of Algorithms. Within Yahoo, the pKMV and Alpha algorithms are used widely. In particular, stream cardinalities in Yahoo empirically satisfy a power law, with some very large streams and many short ones, and pKMV is an attractive option for such settings. We have released an optimized open-source implementation of our algorithms at `http://datasketches.github.io/`.

2.4 Formal Definition of Theta-Sketch Framework

The Theta-Sketch Framework is defined as follows. This definition is specific to the multi-stream setting where the goal is to output $n_{P,U}$, where $U = \cup_{j=1}^{m} A_j$ is the union of constituent streams $A_1, \ldots, A_m$.

▶ **Definition 1.** The Theta-Sketch Framework consists of the following components:

- The data type (θ, S), where $0 < \theta \leq 1$ is a threshold, and S is the set of all unique hashed stream items $0 \leq x < 1$ that are less than θ. We will generically use the term "theta-sketch" to refer to an instance of this data type.
- The universal "combining function" ThetaUnion(), defined in Algorithm 1, that takes as input a collection of theta-sketches (purportedly obtained by running samp[T]() on constituent streams $A_1, \ldots, A_m$), and returns a single theta-sketch (purportedly of the union stream $U = \cup_{i=1}^{m} A_i$).
- The function EstimateOnSubPopulation(), defined in Algorithm 1, that takes as input a theta-sketch (θ, S) (purportedly obtained from some stream A) and a property $P \subseteq [n]$ and returns an estimate of $\hat{n}_{P,A}$.

Any instantiation of the Theta-Sketch Framework must specify a "threshold choosing function" (TCF), denoted $T(k, A, h)$, that maps a target sketch size, a stream, and a hash function h to a threshold θ. Any TCF T implies a "base" sampling procedure samp[T]() that maps a target size, a stream A, and a hash function to a theta-sketch using the pseudocode shown in Algorithm 1. One can obtain an estimate $\hat{n}_{P,A}$ for $n_{P,A}$ by feeding the resulting theta-sketch into EstimateOnSubPopulation().

Given constituent streams $A_1, \ldots, A_m$, the instantiation obtains an estimate $\hat{n}_{P,U}$ of $n_{P,U}$ by running samp[T]() on each constituent stream A_j, feeding the resulting theta-sketches to ThetaUnion() to obtain a "combined" theta-sketch for $U = \cup_{i=1}^{m} A_i$, and then running EstimateOnSubPopulation() on this combined sketch.

▶ Remark. Definition 1 assumes for simplicity that the same TCF T is used in the base sampling algorithms run on each of the constituent streams. However, all of our results that depend only on 1-Goodness (*e.g.* unbiasedness of estimates and non-correlation of "per-item estimates") hold even if different 1-Good TCF's are used on each stream, and even if different values of k are employed.

2.5 Summary of Contributions

In summary, our contributions are: (1) Formulating the Theta-Sketch Framework. (2) Identifying a mild technical condition (1-Goodness) on TCF's ensuring that the framework's estimators are unbiased. (3) Identifying an additional mild technical condition (monotonicity) ensuring that the framework's estimators come with strong variance bounds analogous to Equation (1). (4) Introducing the pKMV Algorithm, a novel variant of multiKMV that can be useful in industrial big-data systems. (5) Proving that multiKMV, multiAdapt, and pKMV all satisfy 1-Goodness and monotonicity, implying unbiasedness and variance bounds for each. (6) Introducing the Alpha Algorithm, and proving that it satisfies 1-Goodness (thus implying unbiasedness), but not monotonicity. We also derive quantitative bounds on the Alpha Algorithm's variance in the single-stream setting, and present experimental evidence that it provides a novel tradeoff between accuracy, space usage, update speed, and applicability in both the single-stream and multi-stream settings.

3 Analysis of the Theta-Sketch Framework

Section Outline. Section 3.1 shows that KMV and Adaptive Sampling are both instantiations of the Theta-Sketch Framework. Section 3.2 defines 1-Goodness. Section 3.3 proves that the TCF's that instantiate behavior identical to KMV and Adapt both satisfy 1-Goodness. Section 3.4 proves that if a framework instantiation's TCF satisfies 1-Goodness, then so does the TCF that is implicitly applied to the union stream via the composition of the instantiation's base algorithm and the function ThetaUnion(). Section 3.5 proves that the estimator $\hat{n}_{P,A}$ for $n_{P,A}$ returned by EstimateOnSubPopulation() is unbiased when applied to any theta-sketch produced by a TCF satisfying 1-Goodness. Section 3.6 defines monotonicity and shows that 1-Goodness and monotonicity together imply variance bounds on $\hat{n}_{P,U}$. Section 3.7 explains how to tweak the Theta-Sketch Framework to handle set intersections and other set operations on streams.

3.1 Example Instantiations

Define m_{k+1} to be the $k{+}1^{\text{st}}$ smallest unique hash value in $h(A)$ (the hashed version of the input stream). The following is an easy observation.

▶ **Observation 2.** When the Theta-Sketch Framework is instantiated with the TCF $T(k, A, h) = m_{k+1}$, the resulting instantiation is equivalent to the multiKMV algorithm outlined in Section 2.2.

Let β be any real value in $(0, 1)$. For any z, define $\beta^{i(z)}$ to be the largest value of β^i (with i a non-negative integer) that is less than z.

▶ **Observation 3.** When the Theta-Sketch Framework is instantiated with the TCF $T(k, A, h) = \beta^{i(m_{k+1})}$ the resulting instantiation is equivalent to multiAdapt, which combines Adaptive Sampling with a growing union rule (cf. Section 2.2).[4]

3.2 Definition of 1-Goodness

The following circularity is a main source of technical difficulty in analyzing theta sketches: for any given identifier ℓ in a stream A, whether its hashed value $x_\ell = h(\ell)$ will end up in a sketch's sample set S depends on a comparison of x_ℓ versus a threshold $T(X^{n_A})$ that depends on x_ℓ itself. Adapting a technique from [5], we partially break this circularity by analyzing the following infinite family of projections of a given threshold choosing function $T(X^{n_A})$.

▶ **Definition 4** (Definition of Fix-All-But-One Projection)**.** Let T be a threshold choosing function. Let ℓ be one of the n_A unique identifiers in a stream A. Let $X^{n_A}_{-\ell}$ be a fixed assignment of hash values to all unique identifiers in A *except* for ℓ. Then the fix-all-but-one projection $T_\ell[X^{n_A}_{-\ell}](x_\ell) : (0, 1) \to (0, 1]$ of T is the function that maps values of x_ℓ to theta-sketch thresholds via the definition $T_\ell[X^{n_A}_{-\ell}](x_\ell) = T(X^{n_A})$, where X^{n_A} is the obvious combination of $X^{n_A}_{-\ell}$ and x_ℓ.

[5] analyzed similar projections under the assumption that the base algorithm is specifically (a weighted version of) KMV; we will instead impose the weaker condition that every fix-all-but-one projection satisfies 1-Goodness, defined below.[5]

---

[4] Section 2.2 assumed that the parameter β was set to the most common value: $1/2$.

[5] We chose the name 1-Goodness due to the reference to Fix-All-But-*One* Projections.

▶ **Definition 5** (Definition of 1-Goodness for Univariate Functions). A function $f(x) : (0,1) \to (0,1]$ satisfies 1-Goodness iff there exists a fixed threshold F such that:

$$\text{If } x < F, \text{ then } f(x) = F. \tag{2}$$

$$\text{If } x \geq F, \text{ then } f(x) \leq x. \tag{3}$$

▶ **Condition 6** (Definition of 1-Goodness for TCF's). A TCF $T(X^{n_A})$ satisfies 1-Goodness iff for every stream A containing n_A unique identifiers, every label $\ell \in A$, and every fixed assignment $X^{n_A}_{-\ell}$ of hash values to the identifiers in $A \setminus \ell$, the fix-all-but-one projection $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell)$ satisfies Definition 5.

3.3 TCF's of multiKMV and multiAdapt Both Satisfy 1-Goodness

The following two easy theorems show that the Threshold Choosing Functions used respectively in KMV and in Adaptive Sampling both satisfy the 1-Goodness condition.

▶ **Theorem 7.** *If $T(X^{n_A}) = m_{k+1}$, then every fix-all-but-one projection $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell)$ of T satisfies 1-Goodness.*

Proof. Let $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell)$ be any specific fix-all-but-one-projection of $T(X^{n_A}) = m_{k+1}$. We will exhibit the fixed value $F_\ell[X^{n_A}_{-\ell}]$ that causes (2) and (3) to be true for this projection. Let a and b respectively be the k'th and $(k{+}1)^{\text{st}}$ smallest hash values in $X^{n_A}_{-\ell}$. Then Subconditions (2) and (3) hold for $F_\ell[X^{n_A}_{-\ell}] = a$. There are three cases:

Case $(x_\ell < a < b)$: In this case, $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = T(X^{n_A}) = m_{k+1} = a$. Since $x_\ell < (F_\ell[X^{n_A}_{-\ell}] = a)$, (2) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = a) = F_\ell[X^{n_A}_{-\ell}]$, and (3) holds vacuously.

Case $(a < x_\ell < b)$: In this case, $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = T(X^{n_A}) = m_{k+1} = x_\ell$. Since $x_\ell \geq (F_\ell[X^{n_A}_{-\ell}] = a)$, (3) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = x_\ell) \leq x_\ell$, and (2) holds vacuously.

Case $(a < b < x_\ell)$: In this case, $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = T(X^{n_A}) = m_{k+1} = b$. Since $x_\ell \geq (F_\ell[X^{n_A}_{-\ell}] = a)$, (3) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = b) < x_\ell$, and (2) holds vacuously.

◀

▶ **Theorem 8.** *If $T(X^{n_A}) = \beta^{i(m_{k+1})}$, then every fix-all-but-one projection $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell)$ of T satisfies 1-Goodness.*

Proof. Let $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell)$ be any specific fix-all-but-one-projection of $T(X^{n_A}) = \beta^{i(m_{k+1})}$. We will exhibit the fixed value $F_\ell[X^{n_A}_{-\ell}]$ that causes (2) and (3) to be true for this projection. Let a and b respectively be the k'th and $(k{+}1)^{\text{st}}$ smallest hash values in $X^{n_A}_{-\ell}$. Then Subconditions (2) and (3) hold for $F_\ell[X^{n_A}_{-\ell}] = \beta^{i(a)}$. There are four cases:

Case $(x_\ell < \beta^{i(a)} < a < b)$: $m_{k+1} = a$, so $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(a)}$. Since $x_\ell < F_\ell[X^{n_A}_{-\ell}] = \beta^{i(a)}$, (2) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(a)}) = F_\ell[X^{n_A}_{-\ell}]$, and (3) holds vacuously.

Case $(\beta^{i(a)} < x_\ell < a < b)$: $m_{k+1} = a$, so $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(a)}$. Since $x_\ell \geq F_\ell[X^{n_A}_{-\ell}] = \beta^{i(a)}$, (3) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(a)}) < x_\ell$, and (2) holds vacuously.

Case $(\beta^{i(a)} < a < x_\ell < b)$: $m_{k+1} = x_\ell$, so $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(x_\ell)}$. Since $x_\ell \geq F_\ell[X^{n_A}_{-\ell}] = \beta^{i(a)}$, (3) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(x_\ell)}) < x_\ell$, and (2) holds vacuously.

Case $\beta^{i(a)} < a < b < x_\ell)$: $m_{k+1} = b$, so $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(b)}$. Since $x_\ell \geq F_\ell[X^{n_A}_{-\ell}] = \beta^{i(a)}$, (3) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(b)}) < b < x_\ell$, and (2) holds vacuously.

◀

3.4 1-Goodness Is Preserved by the Function ThetaUnion()

Next, we show that if a framework instantiation's TCF T satisfies 1-Goodness, then so does the TCF T^U that is implicitly being used by the theta-sketch construction algorithm defined by the composition of the instantiation's base sampling algorithms and the function ThetaUnion(). We begin by formally extending the definition of a fix-all-but-one projection to cover the degenerate case where the label ℓ isn't actually a member of the given stream A.

▶ **Definition 9.** Let A be a stream containing n_A identifiers. Let ℓ be a label that is *not* a member of A. Let the notation $X^{n_A}_{-\ell}$ refer to an assignment of hash value to *all* identifiers in A. For any hash value x_ℓ of the non-member label ℓ, define the value of the "fix-all-but-one" projection $T_\ell[X^{n_A}_{-\ell}](x_\ell)$ to be the constant $T(X^{n_A}_{-\ell})$.

▶ **Theorem 10.** *If the threshold choosing functions $T^{(j)}(X^{n_{A_j}})$ of the base algorithms used to create sketches of m streams A_j all satisfy Condition 6, then so does the TCF:*

$$T^U(X^{n_U}) = \min_j\{T^{(j)}(X^{n_{A_j}})\} \tag{4}$$

that is implicitly applied to the union stream via the composition of those base algorithms and the procedure ThetaUnion*().*

Proof. Let $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$ be any specific fix-all-but-one projection of the threshold choosing function $T^U(X^{n_U})$ defined by Equation (4). We will exhibit the fixed value $F^U[X^{n_U}_{-\ell}]$ that causes (2) and (3) to be true for $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$.

The projection $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$ is specified by a label $\ell \in (A_U = \cup_j A_j)$, and a set $X^{n_U}_{-\ell}$ of fixed hash values for the identifiers in $A_U \backslash \ell$. For each j, those fixed hash values $X^{n_U}_{-\ell}$ induce a set $X^{n_{A_j}}_{-\ell}$ of fixed hash values for the identifiers in $A_j \backslash \ell$. The combination of ℓ and $X^{n_{A_j}}_{-\ell}$ then specifies a projection $\mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)$ of $T^{(j)}(X^j)$. Now, if $\ell \in A_j$, this is a fix-all-but-one projection according to the original Definition 4, and according to the current theorem's pre-condition, this projection must satisfy 1-Goodness for univariate functions. On the other hand, if $\ell \notin A_j$, this is a fix-all-but-one projection according to the extended Definition 9, and is therefore a constant function, and therefore satisfies 1-Goodness. Because the projection $\mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)$ satisfies 1-Goodness either way, there must exist a fixed value $F^j[X^{n_{A_j}}_{-\ell}]$ such that Subconditions (2) and (3) are true for $\mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)$.
We now show that the value $\mathrm{F}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}] := \min_j(\mathrm{F}^{\mathrm{j}}_\ell[X^{n_{A_j}}_{-\ell}])$ causes Subconditions (2) and (3) to be true for the projection $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$, thus proving that this projection satisfies 1-Goodness.

To show: $x_\ell < \mathrm{F}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}]$ implies $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell) = \mathrm{F}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}]$. The condition $x_\ell < \mathrm{F}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}]$ implies that for all j, $x_\ell < \mathrm{F}^{\mathrm{j}}_\ell[X^{n_{A_j}}_{-\ell}]$. Then, for all j, $\mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell) = \mathrm{F}^{\mathrm{j}}_\ell[X^{n_{A_j}}_{-\ell}]$ by Subcondition (2) for the various $\mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)$. Therefore, $\mathrm{F}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}] = \min_j(\mathrm{F}^{\mathrm{j}}_\ell[X^{n_{A_j}}_{-\ell}]) = \min_j(\mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)) = \mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$, where the last step is by Eqn (4). This establishes Subcondition (2) for the projection $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$.

To show: $x_\ell \geq \mathrm{F}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}]$ implies $x_\ell \geq \mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$. Because x_ℓ is greater than or equal to $\mathrm{F}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}] = \min_j(\mathrm{F}^{\mathrm{j}}_\ell[X^{n_{A_j}}_{-\ell}])$, there exists a j such that $x_\ell \geq \mathrm{F}^{\mathrm{j}}_\ell[X^{n_{A_j}}_{-\ell}]$. By Subcondition (3) for this $\mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)$, we have $x_\ell \geq \mathrm{T}^{(\mathrm{j})}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)$. By Eqn (4), we then have $x_\ell \geq \mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$, thus establishing Subcondition (3) for $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$.

Finally, because the above argument applies to every projection $\mathrm{T}^{\mathrm{U}}_\ell[X^{n_U}_{-\ell}](x_\ell)$ of $T^U(X^{n_U})$, we have proved the desired result that $T^U(X^{n_U})$ satisfies Condition 6. ◀

3.5 Unbiasedness of EstimateOnSubPopulation()

We now show that 1-Goodness of a TCF implies that the corresponding instantiation of the Theta-Sketch Framework provides unbiased estimates of the number of unique identifiers on a stream or on the union of multiple streams.

▶ **Theorem 11.** *Let A be a stream containing n_A unique identifiers, and let P be a property evaluating to 1 on an arbitrary subset of the identifiers. Let h denote a random hash function. Let T be a threshold choosing function that satisfies Condition 6. Let (θ, S_A) denote a sketch of A created by $\text{samp}[T](k, A, h)$, and as usual let $P(S_A)$ denote the subset of hash values in S_A whose corresponding identifiers satisfy P. Then $\mathrm{E}_h(\hat{n}_{P,A}) := \mathrm{E}_h\left(\frac{|P(S_A)|}{\theta}\right) = n_{P,A}$.*

Theorems 10 and 11 together imply that, in the multi-stream setting, the estimate $\hat{n}_{P,U}$ for $n_{P,U}$ output by the Theta-Sketch Framework is unbiased, assuming the base sampling schemes $\text{samp}_j()$ each use a TCF $T^{(j)}$ satisfying 1-Goodness.

Proof. Let A be a stream, and let T be a Threshold Choosing Function that satisfies 1-Goodness. Fix any $\ell \in A$. For any assignment X^{n_A} of hash values to identifiers in A, define the "per-identifier estimate" V_ℓ as follows:

$$V_\ell(X^{n_A}) = \frac{S_\ell(X^{n_A})}{T(X^{n_A})} \quad \text{where} \quad S_\ell(X^{n_A}) = \begin{cases} 1 \text{ if } x_\ell < T(X^{n_A}) \\ 0 \text{ otherwise.} \end{cases} \tag{5}$$

Because T satisfies 1-Goodness, there exists a fixed threshold $F(X^{n_A}_{-\ell})$ for which it is a straightforward exercise to verify that:

$$V_\ell(X^{n_A}) = \begin{cases} 1/F(X^{n_A}_{-\ell}) \text{ if } x_\ell < F(X^{n_A}_{-\ell}) \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

Now, conditioning on $X^{n_A}_{-\ell}$ and taking the expectation with respect to x_ℓ:

$$E(V_\ell | X^{n_A}_{-\ell}) = \int_0^1 V_\ell[X^{n_A}](x_\ell)dx_\ell = F(X^{n_A}_{-\ell}) \cdot \frac{1}{F(X^{n_A}_{-\ell})} = 1. \tag{7}$$

Since Equation (7) establishes that $E(V_\ell) = 1$ when conditioned on each $X^{n_A}_{-\ell}$, we also have $E(V_\ell) = 1$ when the expectation is taken over all X^{n_A}. By linearity of expectation, we conclude that $E(\hat{n}_{P,A}) = \sum_{\ell \in A : P(\ell)=1} E(V_\ell) = n_{P,A}$. ◀

3.6 1-Goodness and Monotonicity Imply Variance Bound

As usual, let $U = \cup_{i=1}^m A_i$ be the union of m data streams. Our goal in this section is to identify conditions on a threshold choosing function which guarantee the following: whenever the Theta-Sketch Framework is instantiated with a TCF T satisfying the conditions, then for any property $P \subseteq [n]$, the variance $\sigma^2(\hat{n}_{P,U})$ of the estimator obtained from the Theta-Sketch Framework is bounded above by the variance of the estimator obtained by running $\text{samp}[T]()$ on the stream $A^* := A_1 \circ A_2 \circ \cdots \circ A_m$ obtained by concatenating $A_1, \ldots, A_m$.

It is easy to see that 1-Goodness alone is not sufficient to ensure such a variance bound. Consider, for example, a TCF T that runs KMV on a stream A unless it determines that $n_A \geq C$, for some fixed value C, at which point it sets θ to 1 (thereby causing $\text{samp}[T]()$ to sample all elements from A). Note that such a base sampling algorithm is not implementable by a sublinear space streaming algorithm, but T nonetheless satisfies 1-Goodness. It is easy to see that such a base sampling algorithm will fail to satisfy our desired comparative

variance result when run on constituent streams $A_1, \ldots, A_m$ satisfying $n_{A_i} < C$ for all i, and $n_U > C$. In this case, the variance of $\hat{n}_U$ will be positive, while the variance of the estimator obtained by running samp$[T]$ directly on A^* will be 0.

Thus, for our comparative variance result to hold, we assume that T satisfies both 1-Goodness and the following additional monotonicity condition.

▶ **Condition 12** (Monotonicity Condition)**.** Let A_0, A_1, A_2 be any three streams, and let $A^* := A_0 \circ A_1 \circ A_2$ denote their concatenation. Fix any hash function h and parameter k. Let $\theta = T(k, A_1, h)$, and $\theta' = T(k, A^*, h)$. Then $\theta' \leq \theta$.

▶ **Theorem 13.** *Suppose that the Theta-Sketch Framework is instantiated with a TCF T that satisfies Condition 6 (1-Goodness), as well as Condition 12 (monotonicity). Fix a property P, and let $A_1, \ldots A_m$, be m input streams. Let $U = \cup A_j$ denote the union of the distinct labels in the input streams. Let $A^* = A_1 \circ A_2 \circ \ldots \circ A_m$ denote the concatenation of the input streams. Let $(\theta^*, S^*) =$ samp$[T](k, A^*, h)$, and let $\hat{n}^{A^*}_{P,A^*}$ denote the estimate of $n_{P,A^*} = n_{P,U}$ obtained by evaluating* EstimateOnSubPopulation$((\theta^*, S^*), P)$*. Let $(\theta^U, S^U) =$ ThetaUnion$(\{(\theta_j, S_j)\})$, and let $\hat{n}^U_{P,U}$ denote the estimate of $n_{P,U} = n_{P,A^*}$ obtained by evaluating* EstimateOnSubPopulation$((\theta^U, S^U), P)$*. Then, with the randomness being over the choice of hash function h, $\sigma^2(\hat{n}^U_{P,U}) \leq \sigma^2(\hat{n}^{A^*}_{P,A^*})$.*

The proof of Theorem 13 is rather involved, and is deferred to the full version of the paper.

On the applicability of Theorem 13. It is easy to see that Condition 12 holds for any TCF that is (1) order-insensitive and (2) has the property that adding another distinct item to the stream cannot increase the resulting threshold θ. The TCF T used in multiKMV (namely, $T(k, A, h) = m_{k+1}$), satisfies these properties, as does the TCF used in Adaptive Sampling. Since we already showed that both of these TCF's satisfy 1-Goodness, Theorem 13 applies to multiKMV and multiAdapt. In Section 4, we introduce the pKMV algorithm, which is useful in multi-stream settings where the distribution of stream lengths is highly skewed, and we show that Theorem 13 applies to this algorithm as well.

In Section 5, we introduce the Alpha Algorithm and show that it satisfies 1-Goodness. While the Alpha Algorithm does not satisfy monotonicity in general, it *does* under the promise that $A_1, \ldots, A_m$ are pairwise disjoint; Theorem 13 applies in this case. Our experiments (deferred to the full version of the paper) suggest that, in practice, the normalized variance in the multi-stream setting is not much larger than in the pairwise disjoint case.

3.7 Handling Set Intersections

The Theta-Sketch Framework can be tweaked in a natural way to handle set intersection and other set operations, just as was the case for multiKMV. Specifically, define $\theta_U = \min_{j=1}^m \theta_j$, and $S_I = \{(x \in \cap_j S_j) < \theta_U\}$. The estimator for $n_{P,I}$ is $\hat{n}_{P,I} := |P(S_I)|/\theta_U$.

It is not difficult to see that $\hat{n}_{P,I}$ is exactly equal to $\hat{n}_{P',U}$, where P' is the property that evaluates to 1 on an identifier if and only if the identifier satisfies P and is also in I. Since the latter estimator was already shown to be unbiased with variance bounded as per Theorem 13, $\hat{n}_{P,I}$ satisfies the same properties.

4 The pKMV Variant of KMV

Motivation. An internet company involved in online advertising typically faces some version of the following problem: there is a huge stream of events representing visits of users to

web pages, and a huge number of relevant "profiles", each defined by the combination of a predicate on users and a predicate on web pages. On behalf of advertisers, the internet company must keep track of the count of distinct users who generate events that match each profile. The distribution (over profiles) of these counts typically is highly skewed and covers a huge dynamic range, from hundreds of millions down to just a few.

Because the summed cardinalities of all profiles is huge, the brute force technique (of maintaining, for each profile, a hash table of distinct user ids) would use an impractical amount of space. A more sophisticated approach would be to run multiKMV, treating each profile as separate stream A_i. This effectively replaces each hash table in the brute force approach with a KMV sketch. The problem with multiKMV in this setting is that, while KMV does avoid storing the entire data stream for streams containing more than k distinct identifiers, KMV produces no space savings for streams shorter than k. Because the vast majority of profiles contain only a few users, replacing the hash tables in the brute force approach by KMV sketches might still use an impractical amount of space.

On the other hand, fixed-threshold sampling with $\theta = p$ for a suitable sampling rate p, would always result in an expected factor $1/p$ saving in space, relative to storing the entire input stream. However, this method may result in too large a sample rate for long streams (i.e., for profiles satisfied by many users), also resulting in an impractical amount of space.

The pKMV algorithm. In this scenario, the hybrid Threshold Choosing Function $T(k, A, h) = \min(m_{k+1}, p)$ can be a useful compromise, as it ensures that even short streams get downsampled by a factor of p, while long streams produce at most k samples. While it is possible to prove that this TCF satisfies 1-Goodness via a direct case analysis, the property can also be established by an easier argument: Consider a hypothetical computation in which the ThetaUnion procedure is used to combine two sketches of the same input stream: one constructed by KMV with parameter k, and one constructed by fixed-threshold sampling with parameter p. Clearly, this computation outputs $\theta = \min(m_{k+1}, p)$. Also, since KMV and fixed-threshold sampling both satisfy 1-Goodness, and ThetaUnion preserves 1-Goodness (cf. Theorem 11), T also satisfies 1-Goodness.

It is easy to see that Condition 12 applies to $T(k, A, h) = \min(m_{k+1}, p)$ as well. Indeed, T is clearly order-insensitive, so it suffices to show that adding an additional identifier to the stream cannot increase the resulting threshold. Since p never changes, the only way that adding another distinct item to the stream could increase the threshold would be by increasing m_{k+1}. However, that cannot happen.

5 Alpha Algorithm

5.1 Motivation and Comparison to Prior Art

Section 3's theoretical results are strong because they cover such a wide class of base sampling algorithms. In fact, 1-Goodness even covers base algorithms that lack certain traditional properties such as invariance to permutations of the input, and uniform random sampling of the input. We are now going to take advantage of these strong theoretical results for the Theta Sketch Framework by devising a novel base sampling algorithm that lacks those traditional properties, but still satisfies 1-Goodness. Our main purpose for describing our Alpha Algorithm in detail is to exhibit the generality of the Theta-Sketch Framework. Nonetheless the Alpha Algorithm does have the following advantages relative to HLL, KMV, and Adaptive Sampling.

Advantages over HLL. Unlike HLL, the Alpha Algorithm provides unbiased estimates for $\textsc{Distinct}_P$ queries for non-trivial predicates P. Also, when instantiating the Theta-Sketch Framework via the Alpha Algorithm in the multi-stream setting, the error behavior scales better than HLL for general set operations (cf. Section 2.2). Finally, because the Alpha Algorithm computes a sample, its output is human-interpretable and amenable to post-processing.

Advantages over KMV. Implementations of KMV must either use a heap data structure or quickselect [12] to give quick access to the $k+1^{\text{st}}$ smallest unique hash value seen so far. The heap-based implementation yields $O(\log k)$ update time, and quickselect, while achieving $O(1)$ update time, hides a large constant factor in the Big-Oh notation (cf. Section 2.2). The Alpha Algorithm avoids the need for a heap or quickselect, yielding superior practical performance.

Advantages over Adaptive Sampling. The accuracy of Adaptive Sampling oscillates as n_A increases. The Alpha Algorithm avoids this behavior.

The remainder of this section provides a detailed analysis of the Alpha Algorithm. In particular, we show that it satisfies 1-Goodness, and we give quantitative bounds on its variance in the single-stream setting. The full version of the paper describes experiments showing that, in both the single- and multi-stream settings, the Alpha Algorithm achieves a novel tradeoff between accuracy, space usage, update speed, and applicability.

5.2 AlphaTCF

Algorithm 2 describes the threshold choosing function AlphaTCF. AlphaTCF can be viewed as a tightly interleaved combination of two different processes. One process uses the set D to remove duplicate items from the raw input stream; the other process uses a technique similar to Approximate Counting [14] to estimate the number of items in the de-duped stream created by the first process. In addition, the second process maintains and frequently reduces a threshold $\theta = \alpha^i$ that is used by the first process to identify hash values that *cannot* be members of S, and therefore don't need to be placed in the de-duping set D. If the set D is implemented using a standard dynamically-resized hash table, then well-known results imply that the amortized cost[6] of processing each stream element is $O(1)$, and the space occupied by the hash table is $O(|D|)$. However, there is a simple optimized implementation of the Alpha Algorithm, based on Cuckoo Hashing, that implicitly, and at zero cost, deletes all members of D that are not less than θ, and therefore are not members of S (see the full version of the paper for details). This does not affect correctness, because those deleted members will not be needed for future de-duping tests of hash values that will all be less than θ. Furthermore, in Theorem 15 below, it is proved that $|S|$ is tightly concentrated around k. Hence, the space usage of this optimized implementation is $O(k)$ with probability $1 - o(1)$.

5.3 AlphaTCF Satisfies 1-Goodness

We will now prove that AlphaTCF satisfies 1-Goodness, thus implying unbiasedness.

▶ **Theorem 14.** *If $T(X^{n_A}) = AlphaTCF$, then every fix-all-but-one projection $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell)$ of $T(X^{n_A})$ satisfies 1-Goodness.*

[6] Recent theoretical results imply that the update time can be made worst-case $O(1)$ [1, 2].

Algorithm 2 The Alpha Algorithm's Threshold Choosing Function

```
1: Function AlphaTCF (target size k, stream A, hash function h)
2: α ← k/(k + 1).
3: prefix(h(A)) ← shortest prefix of h(A) containing exactly k unique hash values.
4: suffix(h(A)) ← the corresponding suffix.
5: D ← the set of unique hash values in prefix(h(A)).
6: i ← 0.
7: for all x ∈ suffix(h(A)) do
8:    if x < α^i then
9:       if x ∉ D then
10:         i ← i + 1.
11:         D ← D ∪ {x}.
12:      end if
13:   end if
14: end for
15: return θ ← α^i.
```

Proof. Fix the number of distinct identifiers n_A in A. Consider any identifier ℓ appearing in the stream, and let $x = h(\ell)$ be its hash value. Fix the hash values of all other elements of the sequence of values $X_{-\ell}^{n_A}$. We need to exhibit a threshold F such that $x < F$ implies $T_\ell[X_{-\ell}^{n_A}](x_\ell)(x) = F$ and $x \geq F$ implies $T_\ell[X_{-\ell}^{n_A}](x) \leq x$.

First, if x lies in one of the first $k + 1$ positions in the stream, then $T_\ell[X_{-\ell}^{n_A}](x)$ is a constant independent of x; in this case, F can be set to that constant.

Now for the main case, suppose that ℓ does not lie in one of the first $k + 1$ positions of the stream. Consider a subdivision of the hashed stream into the initial segment preceding $x = h(\ell)$, then x itself, then the final segment that follows x. Because all hash values besides x are fixed in $X_{-\ell}^{n_A}$, during the initial segment, there is a specific number a of times that θ is decreased. When x is processed, θ is decreased either zero or one times, depending on whether $x < \alpha^a$. Then, during the final segment, θ will be decreased a certain number of additional times, where this number depends on whether $x < \alpha^a$. Let b denote the number of additional times θ is decreased if $x < \alpha^a$, and c the number of additional times θ is decreased otherwise. This analysis is summarized in the following table:

Rule	Condition on x	Final value of θ
L	$x < \alpha^a$	α^{a+b+1}
G	$x \geq \alpha^a$	α^{a+c+0}

We prove the theorem using the threshold $F = \alpha^{a+b+1}$. We note that $F = \alpha^{a+b+1} < \alpha^a$, so F and α^a divide the range of x into three disjoint intervals, creating three cases that need to be considered.

Case 1: $x < F < \alpha^a$. In this case, because $x < F$, we need to show that $T_\ell[X_{-\ell}^{n_A}](x) = F$. By Rule L, $T_\ell[X_{-\ell}^{n_A}](x) = \alpha^{a+b+1} = F$.

Case 2: $F \leq x < \alpha^a$. Because $x \geq F$, we need to show that $T_\ell[X_{-\ell}^{n_A}](x) \leq x$. By Rule L, $T_\ell[X_{-\ell}^{n_A}](x) = \alpha^{a+b+1} = F \leq x$.

Case 3: $F < \alpha^a \leq x$. Because $x \geq F$, we need to show that $T_\ell[X_{-\ell}^{n_A}](x) \leq x$. By Rule G, $T_\ell[X_{-\ell}^{n_A}](x) = \alpha^{a+c+0} \leq \alpha^a \leq x$. ◀

5.4 Analysis of Alpha Algorithm on Single Streams

The following two theorems show that the Alpha Algorithm's space usage and single-stream estimation accuracy are quite similar to those of KMV. That means that it is safe to use the Alpha Algorithm as a drop-in replacement for KMV in a sketching-based big-data system,

which then allows the system to benefit from the Alpha Algorithm's low update cost. See the experiments in the full version of the paper for an empirical comparison of these costs.

Random Variables. When Line 15 of Algorithm 2 is reached after processing a randomly hashed stream, the program variable i is governed by a random variable $\mathcal{I}$. Similarly, when Line 3 of Algorithm 1 is subsequently reached, the cardinality of the set S is governed by a random variable $\mathcal{S}$. The following two theorems characterize the distributions of $\mathcal{S}$ and of the Theta Sketch Framework's estimator $\mathcal{S}/(\alpha^{\mathcal{I}})$. Specifically, Theorem 15 shows that the number of elements sampled by the Alpha Algorithm is tightly concentrated around k, and hence its space usage is concentrated around that of KMV. Theorem 16 shows that the variance of the estimate returned by the Alpha Algorithm is very close to that of KMV. Their proofs are deferred to the full version of the paper.

▶ **Theorem 15.** *Let $\mathcal{S}$ denote the cardinality of the set S computed by the Alpha Algorithm's Threshold Choosing Function (Algorithm 2). Then:*

$$\mathrm{E}(\mathcal{S}) = k. \tag{8}$$

$$\sigma^2(\mathcal{S}) < \frac{k}{2} + \frac{1}{4}. \tag{9}$$

▶ **Theorem 16.** *Let $\mathcal{S}$ denote the cardinality of the set S computed by the Alpha Algorithm's Threshold Choosing Function (Algorithm 2). Then:*

$$\sigma^2(\mathcal{S}/(\alpha^{\mathcal{I}})) = \frac{(2k+1)n_A^2 - (k^2+k)(2n_A - 1) - n_A}{2k^2} \tag{10}$$

$$< \frac{n_A^2}{k - \frac{1}{2}}. \tag{11}$$

5.5 Variance of the Alpha Algorithm in the Multi-Stream Setting

The Alpha Algorithm does not satisfy monotonicity (Condition 12) in general, so Theorem 13 does not immediately imply variance bounds in the multi-stream setting. In fact, we have identified contrived examples in the multi-stream setting on which the variance of the Theta-Sketch Framework when instantiated with the TCF of the Alpha Algorithm is slightly larger than the hypothetical estimator obtained by running the Alpha Algorithm on the concatenated stream $A_1 \circ \ldots A_m$ (the worst-case setting appears to be when $A_1 \ldots A_m$ are all permutations of each other).

However, we show in this section that the Alpha Algorithm does satisfy monotonicity under the promise that all constituent streams are pairwise disjoint. This implies the variance guarantees of Theorem 13 do apply to the Alpha Algorithm under the promise that $A_1, \ldots, A_m$ are pairwise disjoint. Our experiments suggest that, in practice, the normalized variance of the Alpha Algorithm in the multi-stream setting is not much larger than in the pairwise disjoint case.

▶ **Theorem 17.** *The TCF computed by the Alpha Algorithm satisfies Condition 12 under the promise that the streams A_1, A_2, A_3 appearing in Condition 12 are pairwise disjoint.*

Proof. Due to space constraints, the proof is deferred to the full version of the paper. ◀

References

1. Yuriy Arbitman, Moni Naor, and Gil Segev. De-amortized cuckoo hashing: Provable worst-case performance and experimental results. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 107–118, 2009. doi:10.1007/978-3-642-02927-1_11.
2. Yuriy Arbitman, Moni Naor, and Gil Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 787–796, 2010. doi:10.1109/FOCS.2010.80.
3. Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, pages 1–10, 2002. doi:10.1007/3-540-45726-7_1.
4. Kevin S. Beyer, Rainer Gemulla, Peter J. Haas, Berthold Reinwald, and Yannis Sismanis. Distinct-value synopses for multiset operations. *Commun. ACM*, 52(10):87–95, 2009. doi:10.1145/1562764.1562787.
5. Edith Cohen and Haim Kaplan. Leveraging discarded samples for tighter estimation of multiple-set aggregates. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, SIGMETRICS/Performance 2009, Seattle, WA, USA, June 15-19, 2009*, pages 251–262, 2009. doi:10.1145/1555349.1555379.
6. Anirban Dasgupta, Kevin Lang, Lee Rhodes, and Justin Thaler. A framework for estimating stream expression cardinalities. *CoRR*, abs/1510.01455, 2015. URL: http://arxiv.org/abs/1510.01455.
7. Nick G. Duffield, Carsten Lund, and Mikkel Thorup. Priority sampling for estimation of arbitrary subset sums. *J. ACM*, 54(6), 2007. doi:10.1145/1314690.1314696.
8. Philippe Flajolet. On adaptive sampling. *Computing*, 43(4):391–400, 1990. doi:10.1007/BF02241657.
9. Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *SPAA*, pages 281–291, 2001. doi:10.1145/378580.378687.
10. Frédéric Giroire. Order statistics and estimating cardinalities of massive data sets. *Discrete Applied Mathematics*, 157(2):406–427, 2009. doi:10.1016/j.dam.2008.06.020.
11. Stefan Heule, Marc Nunkesser, and Alexander Hall. Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In *Joint 2013 EDBT/ICDT Conferences, EDBT'13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 683–692, 2013. doi:10.1145/2452376.2452456.
12. C. A. R. Hoare. Algorithm 65: Find. *Commun. ACM*, 4(7):321–322, July 1961. doi:10.1145/366622.366647.
13. Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 41–52, 2010. doi:10.1145/1807085.1807094.
14. Robert Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, October 1978. doi:10.1145/359619.359627.