A Framework for Estimating Stream Expression Cardinalities

Anirban Dasgputa[1], Kevin Lang[2], Lee Rhodes[3], and Justin Thaler[2]

[1] Indian Institute of Technology, Gandhinagar
[2] Yahoo Labs
[3] Yahoo! Inc

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.

1 Introduction

Consider a telecommunications provider 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 $\textsc{DistinctOnSubPopulation}_P$ problem, or $\textsc{Distinct}_P$ 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

$\textsc{Distinct}_P$ problem that is unbiased. Moreover, its variance should be bounded by that of the individual sensors' sampling procedures.[4]

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 2.4 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 fully 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 $n_{P,A} := \textsc{Distinct}_P(A)$ denote the number of distinct identifiers that appear in A and satisfy P. For brevity, we let n_A denote $\textsc{Distinct}(A)$. 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 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,

[4] 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.

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. [2,5]), 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 $(0,1)$. 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 [15].[5] Due to space constraints, we defer a thorough overview of these algorithms to Appendix A. Here, we briefly review the main concepts and relevant properties of each.

HLL: HyperLogLog Sketches. HLL is a sketching algorithm for the vanilla 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 denotes the k'th smallest hash value.[6] It has been proved by [2], [14], and others, that $E(\text{KMV}_A) = n_A$, and $\sigma^2(\text{KMV}_A) = \frac{n_A^2-(k+1)n_A+n_A}{k-1} < \frac{n_A^2}{k-1}$. Duffield et al. [7] proposed to change they heap-based implementation of the KMV sketching algorithm to an implementation based on quickselect [16]. 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 $P \subseteq [n]$ (Appendix A has details).

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

[5] Algorithms with better asymptotic bit-complexity are known [17], but they do not match the practical performance of the algorithms discussed here. See Appendix A.3.

[6] Some works use the output k/m_k, e.g. [1,2]. We use k/m_{k+1} because it is unbiased, and for consistency with the work of Cohen and Kaplan [5] described below.

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 [9] 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 A. 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 (see Appendix A for details). 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. Cohen and Kaplan [5] proposed the following adaptation of KMV to handle unions of multiple streams. We refer to this algorithm as multiKMV. For each KMV sketch S_j computed from stream A_j, let M_j denote that sketch's value of m_{k+1}. Define $M_U = \min_{j=1}^m M_j$, and $S_U = \{(x \in \cup_j S_j) < M_U\}$. Then n_U is estimated by $\text{multiKMV}_U := |S_U|/M_U$, and $n_{P,U}$ is estimated by $\text{multiKMV}_{P,U} := \text{KMV}_U \cdot |P(S_U)|/|S_U|$. [5] proved that $\text{multiKMV}_{P,U}$ is unbiased and has variance that dominates the variance of the estimator KMV_U that would be obtained by running KMV directly on the union stream:

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

As observed in [5], KMV can be adapted in a similar manner to handle set-intersections (see Section 3.8 or Appendix A for details).

Adaptive Sampling for set operations on streams. Adaptive Sampling can be combined with a growing union rule, just like KMV, to handle set unions and intersections. We refer to this algorithm as multiAdapt. [13] 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 $\text{multiAdapt}_{P,U}$ is unbiased, and satisfies strong variance bounds.

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 $U = \cup_{j=1}^m A_j$. 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

Algorithm 1 Theta Sketch Framework for estimating $n_{P,U}$. The framework is parameterized by choice of TCFs $T^{(j)}(k,A_j,h)$, one for each input stream.

```
**Definition:** Function $\text{samp}_j[T^{(j)}](k,\ A_j,\ h)$
    $\theta_j \leftarrow \mathrm{T}^{(j)}(k, A_j, h)$
    $S_j \leftarrow \{(x \in h(A_j)) < \theta\}$.
    **return** $(\theta_j, S_j)$.
**Definition:** Function ThetaUnion(Theta Sketches $\{(\theta_j, S_j)\}$)
    $\theta_U \leftarrow \min\{\theta_j\}$.
    $S_U \leftarrow \{(x \in (\cup S_j)) < \theta_U\}$.
    **return** $(\theta_U, S_U)$.
**Definition:** Function EstimateOnSubPopulation(Theta Sketch $(\theta, S)$ produced from
  stream $A$, Property $P$ mapping identifiers to $\{0,1\}$)
    **return** $\hat{n}_A := \frac{|P(S)|}{\theta}$.
```

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. We precisely define the framework in Appendix B; here we provide a detailed overview.

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 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 $\text{samp}_j[T^{(j)}]$ 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 $(\theta_j := T^{(j)}(A_j), S)$ 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 $S_U := \{(x \in \cup_j S_j) < \theta_U\}$ (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 $\hat{n}_{P,U} := \hat{n}_U \cdot |P(S_U)|/|S_U|$ (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 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 TCFs 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 variance in the multi-stream setting is not much larger than in the pairwise disjoint case.

2.4 Summary of Contributions

In summary, our contributions are: (1) Formulating the Theta-Sketch Framework. (2) Identifying a mild technical condition (1-Goodness) on TCFs ensuring that the framework's estimators are unbiased. If each TCF also satisfies a monotonicity condition, the framework's estimators come with strong variance bounds analogous to Equation (1). (3) Proving multiKMV, multiAdapt, and pKMV all satisfy 1-Goodness and monotonicity, implying unbiasedness and variance bounds for each. (4) Introducing the Alpha Algorithm, proving that it is unbiased, and establishing quantitative bounds on its variance in the single-stream setting. (5) Experimental results showing that the Alpha Algorithm instantiation achieves a novel tradeoff between accuracy, space usage, update speed, and applicability.

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. Sections 3.3 and 3.4 prove that the TCFs that instantiate behavior identical to KMV and Adapt both satisfy 1-Goodness. Section 3.5 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 3.6 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 3.7 defines monotonicity and shows that 1-Goodness and monotonicity together implies variance bounds on $\hat{n}_{P,U}$. Section 3.8 explains how to tweak the Theta-Sketch Framework to handle set intersections and other set operations on streams. Finally, Section 3.9 describes the pKMV variant of KMV.

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 1 *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 2 *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).*[7]

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 1 (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 $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) : (0, 1) \to (0, 1]$ of T is the function that maps values of x_ℓ to theta-sketch thresholds via the definition $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = T(X^{n_A})$, 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.[8]

Definition 2 (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 3 (Definition of 1-Goodness for TCFs) *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 $A \backslash \ell$, the fix-all-but-one projection $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell)$ satisfies Definition 2.*

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

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

3.3 TCF of multiKMV Satisfies 1-Goodness

The following theorem shows that the TCF used in KMV satisfies 1-Goodness. The proof is in Appendix D.

Theorem 4. *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.*

3.4 TCF of multiAdapt Satisfies 1-Goodness

The following theorem shows that the TCF used in Adaptive Sampling satisfies 1-Goodness. The proof is in Appendix E.

Theorem 5. *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 has a good shape, as specified by Definition 2.*

3.5 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 3. *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 6. *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 3, then so does the TCF:*

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

that is implicitly being 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 1, 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 3, and is therefore a constant function, and therefore has a good shape. Because the projection $\mathrm{T}^{\mathrm{(j)}}_\ell[X^{n_{A_j}}_{-\ell}](x_\ell)$ has a good shape 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 has a good shape.

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 $x_\ell \geq \mathrm{F}^{\mathrm{U}}_{\ell}[X^{n_U}_{-\ell}] = \min_j(\mathrm{F}^{\mathrm{j}}_{\ell}[X^{n_{A_j}}_{-\ell}])$, 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 3.

3.6 Unbiasedness of EstimateOnSubPopulation()

We now show that 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 7. *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 3. Let (θ, S_A) denote a sketch of A created by* $\mathrm{samp}[T](k, A, h)$*, 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 6 and 7 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 $\mathrm{samp}_j()$ all use a TCF T 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, the 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 $E(\hat{n}_{P,A}) = \sum_{\ell \in A : P(\ell)=1} E(V_\ell) = n_{P,A}$.

Is 1-Goodness Necessary for Unbiasedness? Appendix F gives an example showing that 1-Goodness cannot be substantially weakened while still guaranteeing unbiasedness of the estimate $\hat{n}_{P,U}$ returned by the Theta-Sketch Framework.

3.7 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 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 points it sets θ to 1 (thereby causing 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 8 (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 9. *Suppose that the Theta-Sketch Framework is instantiated with a TCF T that satisfies Condition 3 (1-Goodness), as well as Condition 8 (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 $(\theta^*, S^*) = \text{samp}[T](k, A^*, h)$, and let $\hat{n}^{A^*}_{P,A^*}$ denote the corresponding estimate of $n_{P,A^*} = n_{P,U}$ obtained by feeding (θ^*, S^*) into* EstimateOnSubPopulation*(). 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 feeding (θ^U, S^U) into* EstimateOnSubPopulation*(). 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 9 is rather involved, and is in Appendix G.

On the applicability of Theorem 9. It is easy to see that Condition 8 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 TCFs satisfy 1-Goodness, Theorem 9 applies to multiKMV and multiAdapt. In Section 3.9, 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 9 applies to this algorithm as well.

In Section 4, we introduce the Alpha Algorithm and show that it satisfies 1-Goodness. Unfortunately, the Alpha Algorithm does not satisfy monotonicity in general. The algorithm does, however, satisfy monotonicity under the promise that $A_1, \ldots, A_m$ are pairwise disjoint, and Theorem 9 applies in this case. Our experiments (Appendix J) suggest that, in practice, the variance in the multi-stream setting is not much larger than in the pairwise disjoint case.

3.8 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 (cf. Section A.2). Specifically, as in the set-union case, define $\theta_U = \min_{j=1}^{m} \theta_j$, and $S_U = \{(x \in \cup_j S_j) < \theta_j\}$. In addition, define $S_I = \{(x \in \cap_j S_j) < \theta_U\}$. The estimator for $n_{P,I}$ is $\hat{n}_{P,I} := \hat{n}_U \cdot |P(S_I)|/|S_U|$.

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 9, $\hat{n}_{P,I}$ satisfies the same properties.

3.9 The pKMV Variant of KMV

In multi-stream settings where the lengths of constituent streams are highly skewed, the Theta Choosing Function $T(k, A, h) = \min(m_{k+1}, p)$ can be useful, where p is a user-specified constant. Due to space constraints, a detailed motivation for this choice of TCF is deferred to Appendix H. In short, it is useful because it ensures that even very short streams get downsampled by a factor of p, while long streams produce at most k samples. In Appendix H, we show that T satisfies both 1-Goodness and monotonicity.

4 Alpha Algorithm

We defer a detailed description of the Alpha Algorithm to Appendix I. Here, we briefly describe its advantages over HLL, KMV, and Adaptive Sampling.

Advantages over HLL. Unlike HLL, the Alpha Algorithm provides unbiased estimates for 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 [16] to give quick access to the k'th smallest 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.

Appendix I 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. Appendix **??** describes experiments showing that, in both the single- and multi-stream settings, the algorithm achieves a novel tradeoff between accuracy, space usage, update speed, and applicability.

References

1. Z. Bar-Yossef, T. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan, "Counting distinct elements in a data stream," in *RANDOM*, pp. 1–10, 2002.
2. K. Beyer, R. Gemulla, P. J. Haas, B. Reinwald, and Y. Sismanis, "Distinct-value synopses for multiset operations," *CACM*, vol. 52, no. 10, pp. 87–95, 2009.
3. A. Chen, J. Cao, and T. Bu, "A simple and efficient estimation method for stream expression cardinalities," in *VLDB*, pp. 171–182, 2007.
4. E. Cohen, "All-distances sketches, revisited: HIP estimators for massive graphs analysis," in *PODS*, pp. 88–99, 2014.
5. E. Cohen and H. Kaplan, "Leveraging discarded samples for tighter estimation of multiple-set aggregates," in *SIGMETRICS*, pp. 251–262, 2009.
6. E. Cohen and H. Kaplan, "Summarizing data using bottom-k sketches," in *PODC*, pp. 225–234, 2007.
7. N. G. Duffield, C. Lund, and M. Thorup, "Priority sampling for estimation of arbitrary subset sums," *J. ACM*, vol. 54, no. 6, 2007.
8. P. Flajolet, "Approximate counting: a detailed analysis," *BIT Numerical Mathematics*, vol. 25, no. 1, pp. 113–134, 1985.
9. P. Flajolet, "On adaptive sampling," *Computing*, vol. 43, no. 4, pp. 391–400, 1990.
10. P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier, "Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm," *DMTCS Proceedings*, 2008.
11. P. Flajolet and G. Martin, "Probabilistic counting algorithms for data base applications," *JCSS*, vol. 31, no. 2, pp. 182–209, 1985.
12. S. Ganguly, M. Garofalakis, and R. Rastogi, "Processing set expressions over continuous update streams," in *SIGMOD*, pp. 265–276, 2003.
13. P. B. Gibbons and S. Tirthapura, "Estimating simple functions on the union of data streams," in *SPAA*, pp. 281–291, 2001.
14. F. Giroire, "Order statistics and estimating cardinalities of massive data sets," *Discrete Applied Mathematics*, vol. 157, no. 2, pp. 406–427, 2009.
15. S. Heule, M. Nunkesser, A. Hall, "Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm," *EDBT*, p. 683-692, 2013.
16. C. A. R. Hoare, "Algorithm 65: Find," *CACM*, vol. 4, pp. 321–322, July 1961.
17. D. M. Kane, J. Nelson, and D. P. Woodruff, "An optimal algorithm for the distinct elements problem," in *PODS*, pp. 41–52, 2010.
18. R. Morris, "Counting large numbers of events in small registers," *CACM*, vol. 21, no. 10, pp. 840–842, 1978.
19. D. Ting, "Streamed approximate counting of distinct elements: Beating optimal batch methods," in *KDD*, pp. 442–451, 2014.
20. M. Thorup, "Bottom-k and priority sampling, set similarity and subset sums with minimal independence," in *STOC*, pp. 371–380, 2013.

A Detailed Overview of Prior Work

A.1 Algorithms for Single Streams

HLL: HyperLogLog Sketches. HLL is a sketching algorithm for the vanilla Distinct problem. It uses a hash function to randomly distribute the elements of a stream A amongst k buckets. For each bucket i, there is a register b_i, whose length is $O(\log \log n)$ bits, that essentially contains the largest number of leading zeros in the hashed value of any stream element sent to that bucket. For each stream element, this data structure can clearly be updated in $O(1)$ time. The HLL estimator for n_A which we denote HLL_A, is a certain non-linear function of the k bucket values b_i; see [10]. It has been proved by [10] that, as $n_A \to \infty$, $E(\text{HLL}_A) \to\ n_A$, and $\sigma^2(\text{HLL}_A) \to\ 1.04(n_A^2/k)$.

Unlike the KMV and Adaptive Sampling algorithms described below, 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). Qualitatively, the reason that HLL cannot estimate $n_{P,A}$ is that, unlike the other algorithms, HLL does not maintain any kind of sample of identifiers from the stream.

KMV: K'th Minimum Value Sketches. The KMV sketching procedure for estimating $\text{Distinct}(A)$ 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 $\text{Distinct}(A)$ is

$$\text{KMV}_A =\ k/m_{k+1}, \tag{8}$$

where m_k denotes the k'th smallest hash value. It has been proved by [2], [14], and others, that $E(\text{KMV}_A) = n_A$, and

$$\sigma^2(\text{KMV}_A) = \frac{n_A^2 - (k+1)n_A + n_A}{k-1} < \frac{n_A^2}{k-1}. \tag{9}$$

Duffield et al. [7] proposed to change they heap-based implementation of the KMV sketching algorithm to an implementation based on quickselect [16]. This reduces the sketch update cost from $O(\log k)$ to amortized $O(1)$. However, this $O(1)$ has a larger constant factor than that of competing methods.

The KMV sketching procedure can be extended to estimate $n_{P,A}$ for any property $P \subseteq [n]$, as explained below. To accomplish this, the KMV sketch must keep not just the k smallest unique hash values that have been observed in the stream, but also the actual item identifiers corresponding to the hash values.[9] This allows the algorithm to determine which of the items in the sample satisfy the property P, even when P is not known until query time.

Motivated by the identity $n_{P,A} = n_A \cdot (n_{P,A}/n_A)$, the quantity $\text{KMV}_A \cdot est(n_{P,A}/n_A)$ is a plausible estimate of $n_{P,A}$, for any sufficiently accurate estimate

[9] Technically, the sketch need not store the hash values if it stores the corresponding identifiers. Nonetheless, storing the hash values is often desirable in practice, to avoid the need to repeatedly evaluate the hash function.

$est(n_{P,A}/n_A)$ of $n_{P,A}/n_A$. Let S_A denote the k smallest unique hashed values in A, and recall (cf. Section 2.1) that $P(S_A)$ denotes the subset of hash values in S_A whose corresponding identifiers in $[n]$ satisfy the predicate P (the reason we require the sketch to store the actual identifiers that hashed to each value is to allow S_A to be determined from the sketch). Then the fraction $|P(S_A)|/|S_A|$ can serve as the desired estimate of the fraction $n_{P,A}/n_A$. Essentially because S_A is a uniform random sample of A, it has been proved by [2] that the estimate $\mathrm{KMV}_{P,A} = \mathrm{KMV}_A \cdot |P(S_A)|/|S_A|$ of $n_{P,A}$ is unbiased, and has the following variance:

$$\sigma^2(\mathrm{KMV}_{P,A}) = \frac{n_{P,A}((k+1)\, n_A - (k+1)^2 - n_A + k + 1 + n_{P,A})}{(k+1)(k-1)}. \qquad (10)$$

The leading term in this variance expression is $(n_A \cdot n_{P,A})/k$.

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 can be cheaper than KMV. The estimator of n_A is $\mathrm{Adapt}_A = |S|/2^{-i}$. It has been proved by [9] that this estimator is unbiased, and that $\sigma^2(\mathrm{Adapt}_A) \approx 1.44(n_A^2/(k-1))$, 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 A, via $\mathrm{Adapt}_{P,A} = \mathrm{Adapt}_A \cdot |P(S_A)|/|S_A|$. Note that, just as for KMV, this extension requires storing not just the hash values in S, but also the actual identifiers corresponding to each hash value.

Although the stream processing speed of Adaptive Sampling is excellent (see Figure 5), the fact that its accuracy oscillates as n_A increases (see Figures 6 and 7) is a shortcoming of the method.

A.2 Algorithms for Set Operations on Multiple Streams

HLL Sketches for Multiple Streams.

- **Set Union.** A sketch of U can be constructed from m HLL sketches of the A_j's by taking the maximum of the m register values for each of the k buckets. The resulting sketch is identical to an HLL sketch constructed directly from U, so $E(\mathrm{HLL}_U) \to n_U$, and $\sigma^2(\mathrm{HLL}_U) \to 1.04(n_U^2/k)$.
- **Set Intersection.** Given constituent streams $A_1, \ldots, A_m$, the HLL sketch can be extended via the Inclusion/Exclusion (IE) rule to estimate Distinct for various additional set-expressions other than set-union applied to $A_1, \ldots, A_m$. This approach is awkward for complicated expressions, but is straightforward for simple expressions. For example, if $m = 2$, then the HLL+IE estimate of $|I| = |A_1 \cap A_2|$ is $\mathrm{HLL}_{A_1} + \mathrm{HLL}_{A_2} - \mathrm{HLL}_U$.
  Unfortunately, the variance of this estimate is approximately n_U^2/k. This is a factor of n_U^2/n_I^2 larger than the variance of roughly n_I^2/k if one could somehow run HLL directly on I, and 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.

In summary, the main limitations of HLL are its bad error scaling behavior when dealing with set operations other than set-union, as well as the inability to estimate $\textsc{Distinct}_P$ queries for general properties P, even for a single stream A.

multiKMV: KMV for Multiple Streams.

- **Set Union.** For any property P, there are two natural ways to extend KMV to estimate $n_{P,U}$, given a KMV sketch S_j for each constituent stream A_j. The first is to use a "non-growing" union rule, and the second is to use a "growing" union rule (our term).
  With the non-growing union rule, the sketch S_U of U is simply defined to be the set of k smallest hash values in $\cup_{j=1}^{m} S_j$. The resulting sketch is identical to a KMV sketch constructed directly from U, so $E(\mathrm{KMV}_U) = n_U$, and $\sigma^2(\mathrm{KMV}_U) < n_U^2/(k-2)$. Just as the KMV sketch for a single stream A can be adapted to estimated $n_{P,A}$ for any property P, this multi-stream variant of KMV can be adapted to estimate $n_{P,U}$.
  The growing union rule was introduced by Cohen and Kaplan [5]. This rule decreases the variance of estimates for unions and for other set expressions, but also increases the space cost of computing those estimates. Throughout, we refer to Cohen and Kaplan's algorithm as multiKMV.
  For each KMV input sketch S_j, let M_j denote that sketch's value of m_{k+1}. Define $M_U = \min_{j=1}^{m} M_j$, and $S_U = \{(x \in \cup_j S_j) < M_U\}$. Then n_U is estimated by $\mathrm{multiKMV}_U := |S_U|/M_U$, and $n_{P,U}$ is estimated by $\mathrm{multiKMV}_{P,U} := \mathrm{KMV}_U \cdot |P(S_U)|/|S_U|$. [5] proved that $\mathrm{multiKMV}_{P,U}$ is unbiased and has variance that strictly dominates the variance of the estimator KMV_U:
  $$\sigma^2(\mathrm{multiKMV}_{P,U}) \leq \sigma^2(\mathrm{KMV}_U). \tag{11}$$
- **Set Intersection.** multiKMV can be tweaked in a natural way to handle set intersection and other set operations. Specifically, as in the set-union case, define $M_U = \min M_j$, and $S_U = \{(x \in \cup_j S_j) < M_U\}$. In addition, define $S_I = \{(x \in \cap_j S_j) < M_U\}$. The estimator for $n_{P,I}$ is $\mathrm{multiKMV}_{P,I} := \mathrm{KMV}_U \cdot |P(S_I)|/|S_U|$.
  It is not difficult to see that $\mathrm{multiKMV}_I$ is exactly equal to $\mathrm{KMV}_{P',U}$, where $P' = P \cap I$ 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 Equation (1), $\mathrm{multiKMV}_{P,I}$ satisfies the same properties.

multiAdapt: Adaptive Sampling for Multiple Streams.

- **Set Union.** Just as with KMV, for any property P, there are two natural ways to extend Adaptive Sampling to estimate $n_{P,U}$, given an Adaptive Sampling sketch S_j for each constituent stream A_j. The first is to use a non-growing union rule, and the second is to use a growing union rule. For brevity, we will only discuss the growing union rule, as proposed by [13].

We refer to this algorithm as multiAdapt. Let (i_j, S_j) be the sketch of the j'th input stream A_j. The union sketch constructed from these sketches is $(i_U = \max i_j,\ S_U = \{(x \in \cup S_j) < 2^{-i_U}\})$. Then n_U is estimated by $\text{multiAdapt}_U := |S_U|/2^{-i_U}$, and $n_{P,U}$ is estimated by $\text{multiAdapt}_{P,U} := \text{multiAdapt}_U \cdot |P(S_U)|/|S_U|$. [13] 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 in fact both special cases of our Theta-Sketch Framework, and in Section 3 of this paper we will prove (apparently for the first time) that $\text{multiAdapt}_{P,U}$ is unbiased.

- **Set Intersection.** To our knowledge, prior work has not considered extending multiAdapt to handle set operations other than set-union on constituent streams. However, it is possible to tweak multiAdapt in a manner similar to multiKMV to handle these operations.

A.3 Other Related Work

Estimating the number of distinct values for data streams is a well studied problem. The problem of estimating result sizes of set expressions over multiple streams was concretely formulated by Ganguly et al. [12]. Motivated by the question of handling streams containing both insertions and deletions, their construction involves a 2-level hash function that essentially stores a set of counters for each bit-position of an HLL-type hash, and hence is inherently more resource intensive, both in terms of the space and update times.

K'th Minimum Value sketches were introduced by Bar-Yossef et al. [1], and developed into an unbiased scheme that handles set expressions by Beyer et al. [2]. Our own scheme is closely related to the schemes proposed and analyzed in Cohen and Kaplan [5], and in Gibbons and Tirthapura [13]. Chen, Cao and Bu [3] propose a somewhat different scheme for estimating unique counts with set expressions that is based on a data-structure related to the "probabilistic counting" sketches of [11], and also to the multi-bucket KMV sketches of [14] (with $K = 1$). However, the guarantees proved by [3] are asymptotic in nature, and their system's union sketches are the same size as base sketches, and therefore do not provide the increased accuracy that is possible with a "growing" union rule as in [5], in [13], and in this paper's scheme.

Bottom-k sketches [5, 6] are a weighted generalization of KMV that provides unbiased estimates of the weights of arbitrary subpopulations of identifiers. They have small errors even under 2-independent hashing [20]. A closely related method for estimating subpopulation weights is priority sampling [7]. Although this paper's Theta-Sketch Framework offers a broad generalization of KMV, it is not clear that it can support the entire generality of bottom-k sketches for weighted sets.

This paper's "Alpha Algorithm" is inspired by the elegant *Approximate Counting* method of Morris [18], that has previously been applied to the estimation of the frequency moments F_p, for $p \geq 1$. By contrast, *our* task is to estimate $\textsc{Distinct}_P$. The Alpha Algorithm is able to do this because its Approximate

Counting process is tightly interleaved with another process that removes duplicates from the input stream while maintaining a small memory footprint by using feedback from the approximate counter.

Kane et al. [17] gave a streaming algorithm for the DistinctElements problem that outputs a $(1+\epsilon)$-approximation with constant probability, using $\Theta(\epsilon^{-2} + \log(n))$ bits of space. This improves over the bit-complexity of HLL by roughly a $\log \log n$ factor (and avoids the assumption of truly random hash functions). Like HLL, it is not known how to extend the algorithm to handle DistinctOnSubPopulation$_P$ queries for non-trivial properties P, and the algorithm does not appear to have been implemented [15].

In very recent work, Cohen [4] and Ting [19] have proposed new estimators for DistinctElements (called "Historical Inverse Probabililty" (HIP) estimators in [4]). Any sketch which is generated by hashing of each element in the data stream and is not affected by duplicate elements (such as HLL, KMV, Adaptive Sampling, and our Alpha Algorithm) has a corresponding HIP estimator, and [4, 19] show that the HIP estimator reduces the variance of the original sketching algorithm by a factor of 2. However, HIP estimators, in general, can only be computed when processing the stream, and this applies in particular to the HIP estimators of KMV and Adaptive Sampling. Hence, they do not satisfy the mergeablity properties necessary to apply to multi-stream settings.

B Formal Definition of Theta-Sketch Framework

In this section, we consider the Theta-Sketch Framework in the multi-stream setting where the goal is to output $n_{P,U}$, where $U = \cup_{j=1}^{m} A_j$ is the union of constituent streams $A_1, \ldots, A_m$.

Definition 4. *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 4 assumes for simplicity that the same TCF T is used in the base sampling algorithms run in each constituent streams. Our analysis applies even if different TCFs are used on each stream.

C Example Fix-All-But-One Projections

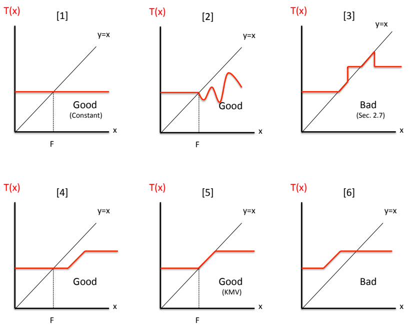


Fig. 1. Six examples of hypothetical projections of TCF's. Four of them satisfy 1-Goodness; the other two do not.

D Proof of Theorem 4

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 $(k{+}1)^{\mathrm{st}}$ 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.

E Proof of Theorem 5

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.

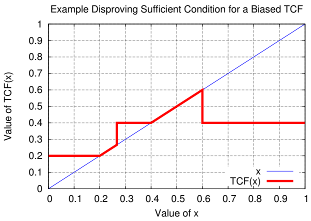
Case $\beta^{i(a)} < a < b < x_\ell):$ $m_{k+1} = b$, so $\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(b)}$. Since $x_\ell \geq (F_\ell[X^{n_A}_{-\ell}] = \beta^{i(a)})$, (3) holds because $(\mathrm{T}_\ell[X^{n_A}_{-\ell}](x_\ell) = \beta^{i(b)}) < b < x_\ell$, and (2) holds vacuously.

F Example Demonstrating 1-Goodness Cannot Be Substantially Weakened

By construction, the following threshold choosing function causes the estimator of the Theta-Sketch Framework to be biased upwards.

$$\mathrm{T}(X^{n_A}) = \begin{cases} m_k \text{ if } \frac{k-1}{m_k} > \frac{k}{m_{k+1}} \\ m_{k+1} \text{ otherwise} \end{cases} \tag{12}$$

Therefore, by the contrapositive of this section's main result, it cannot satisfy Condition 3. It is an interesting exercise to try to establish this fact directly. It can be done by exhibiting a specific target size k, stream A, and partial assignment of hash values $X^{n_A}_{-\ell}$ such that no fixed threshold $F_\ell[X^{n_A}_{-\ell}]$ exists that would satisfy (2) and (3). Here is one such example: $k = 3$, $h(A) = \{0.1, 0.2, 0.4, 0.7, x_\ell\}$.



The non-existence of the required fixed threshold is proved by the above plot of $T(x_\ell)$. The only value of $F_\ell[X^{n_A}_{-\ell}]$ that would satisfy subcondition (2) is 0.2. However, that value does *not* satisfy (3), because $T(x_\ell) > x_\ell$ for $8/30 < x_\ell < 0.4$.

G Proof of Theorem 9

G.1 Proof Overview

The proof introduces the notion of the *fix-all-but-two projection* of a threshold choosing function T. We then introduce a new condition on TCFs that we call 2-Goodness (cf. Appendix G.2). On its face, 2-Goodness may appear to be a stronger requirement than 1-Goodness. However, we show in Section G.3 that this is not the case: 1-Goodness in fact implies 2-Goodness.[10] We show in Appendix G.3 that 2-Goodness implies that "per-identifier estimates" output by the Theta-Sketch Framework are uncorrelated. Finally, in Section G.4, we use this result to complete the proof of Theorem 9.

G.2 Definition of Fix-All-But-Two Projections and 2-Goodness

We begin by defining the Fix-All-But-Two Projection of a TCF.

Definition 5. *Let T be a threshold choosing function and fix a stream A. Let $\ell_1 \neq \ell_2$ be two of the n_A unique identifiers in A. Let $X^{n_A}_{-\ell_1,-\ell_2}$ be a fixed assignment of hash values to all unique identifiers in A* except *for ℓ_1 and ℓ_2. Then the fix-all-but-two projection $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x_{\ell_1}, x_{\ell_2}) : (0,1) \times (0,1) \to (0,1]$ of T is the function that maps values of (x_{ℓ_1}, x_{ℓ_2}) to theta-sketch thresholds via the definition $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x_{\ell_1,\ell_2}) = T(X^{n_A})$, where X^{n_A} is the obvious combination of $X^{n_A}_{-\ell_1,-\ell_2}$, x_{ℓ_1}, and x_{ℓ_2}.*

Next, we define the notion of 2-Goodness for bivariate functions.

Definition 6. *Let $f(x,y) : (0,1) \times (0,1) \to (0,1]$ be a bivariate function. We say that f satisfies 2-Goodness if there exists an $F \in (0,1]$ such that*

– $\max(x,y) < F \Rightarrow f(x,y) = F$.
– $\max(x,y) \geq F \Rightarrow f(x,y) \leq \max(x,y)$.

[10] In fact, the two properties can be shown to be equivalent. We omit the reverse implication, since we will not require it to establish our variance bounds.

Finally we are ready to define 2-Goodness for TCFs.

Condition 10 *A threshold choosing function $T(X^{n_A})$ satisfies 2-Goodness iff for every stream A containing n_A unique identifiers, every pair of identifiers $\ell_1, \ell_2 \in A$, and every fixed assignment $X^{n_A}_{-\ell_1,-\ell_2}$ of hash values to the identifiers in $A \setminus \{\ell_1, \ell_2\}$, the fix-all-but-two projection $\mathrm{T}_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x_{\ell_1}, x_{\ell_2})$ satisfies Definition 6.*

G.3 1-Goodness Implies 2-Goodness

We are ready to show the (arguably surprising) result that if T satisfies 1-Goodness, then it also satisfies 2-Goodness.

Theorem 11. *Let T be a threshold choosing function that satisfies 1-Goodness. Then T also satisfies 2-Goodness.*

Proof. Let $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}]$ be any fix-all-but-two projection of T. Notice that for any $y' \in (0,1)$, $f(x) := T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y')$ is a fix-all-but-one projection of T. Similarly for any $x' \in (0,1)$, $g(y) := T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x', y)$ is a fix-all-but-one-projection of T. Hence, 1-Goodness of T implies the following conditions hold:

Property 1. For all $y' \in (0,1)$, there exists a $G^{y'} \in (0,1]$ such that:

- $x < G^{y'} \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y') = G^{y'}$.
- $x \geq G^{y'} \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y') \leq x$.

Property 2. For all $x' \in (0,1)$, there exists a $H^{x'} \in (0,1]$ such that:

- $y < H^{x'} \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x', y) = H^{x'}$.
- $y \geq H^{x'} \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x', y) \leq y$.

To establish that T satisfies 2-Goodness, we want to prove that there exists an $F \in (0,1]$ such that

- $\max(x,y) < F \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) = F$.
- $\max(x,y) \geq F \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) \leq \max(x,y)$.

We will break the proof down into two lemmas.

Lemma 1. *There exists an $F \in (0,1]$ such that $\max(x,y) < F \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) = F$.*

Proof. By Property 1 above, there exists a $G^0 \in (0,1]$ such that

$$x < G^0 \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, 0) = G^0. \tag{13}$$

Now consider any x in $(0, G^0)$. By Property 2 above, there exists a $H^x \in (0,1]$ such that:

$$y < H^x \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) = H^x. \tag{14}$$

Plugging $y = 0$ into Equation (14) gives $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, 0) = H^x$, while Equation (13) guarantees that $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, 0) = G^0$, so $H^x = G^0$. Substituting G^0 into Equation (14) yields

$$y < G^0 \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) = G^0. \tag{15}$$

Because x was any value in the interval $(0, G^0)$, the lemma is proved with $F = G^0$.

Lemma 2. *The threshold F whose existence was proved in Lemma 1 also has the property that if $\max(x, y) \geq F$, then $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) \leq \max(x, y)$.*

Proof. We start by assuming that $\max(x, y) \geq F$, so at least one of the following must be true: $(x \geq F)$ or $(y \geq F)$. Without loss of generality we will assume that $x \geq F$. By Property 2 above, there exists an $H^x \in (0, 1]$ such that

– $y < H^x \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) = H^x$.
– $y \geq H^x \Rightarrow T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) \leq y$.

Our proof will have two cases, determined by whether $y < H^x$ or $y \geq H^x$.

First case: $y < H^x$. In this case, because $y < H^x$, $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) = H^x$. Also, $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, 0) = H^x$. But $x \geq F = G^0$, so $T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, 0) \leq x$. Putting this all together gives:

$$T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) = H^x = T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, 0) \leq x \leq \max(x, y). \tag{16}$$

Second case: $y \geq H^x$. In this case, because $y \geq H^x$,

$$T_{\ell_1,\ell_2}[X^{n_A}_{-\ell_1,-\ell_2}](x, y) \leq y \leq \max(x, y). \tag{17}$$

1-Goodness Implies Per-Identifier Estimates Are Uncorrelated

Lemma 3. *Fix any stream A, threshold choosing function G, and pair $\ell_1 \neq \ell_2$ in A. Define the "per-identifier estimates" V_{ℓ_1} and V_{ℓ_2} as in Equation* (5)*. Then if T satisfies* 1*-Goodness, the covariance of V_{ℓ_1} and V_{ℓ_2} is 0. In symbols,*

$$\sigma(V_{\ell_1}, V_{\ell_2}) = E_{X^{n_A}}(V_{\ell_1} \cdot V_{\ell_2}) - E_{X^{n_A}}(V_{\ell_1}) \cdot E_{X^{n_A}}(V_{\ell_2}) = 0.$$

Proof. Because T satisfies 1-Goodness, it also satisfies 2-Goodness (cf. Theorem 11), and hence there exists a threshold $F(X^{n_A}_{-\ell_1,-\ell_2})$ for which it is a straightforward exercise to verify that:

$$V_{\ell_1}(X^{n_A}) \cdot V_{\ell_2}(X^{n_A}) = \begin{cases} 1/F(X^{n_A}_{-\ell_1,-\ell_2})^2 \text{ if } \max(x_{\ell_1}, x_{\ell_2}) < F(X^{n_A}_{-\ell_1,-\ell_2}) \\ 0 \text{ otherwise.} \end{cases} \tag{18}$$

Now, conditioning on $X^{n_A}_{-\ell_1,-\ell_2}$ and taking the expectation with respect to pairs (x_{ℓ_1}, x_{ℓ_2}):

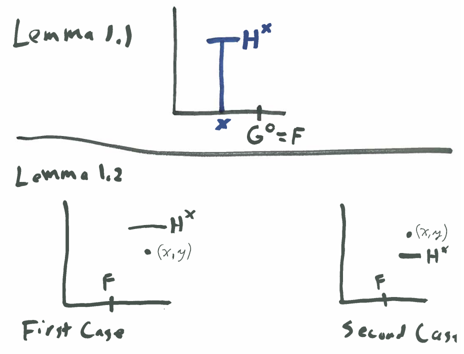


Fig. 2. Some diagrams for Lemmas 1 and 2

$$E(V_{\ell_1} \cdot V_{\ell_2} | X^{n_A}_{-\ell_1,-\ell_2}) = \int_0^1 \int_0^1 V_{\ell_1}(X^{n_A}) V_{\ell_2}(X^{n_A}) dx_{\ell_1} dx_{\ell_2} = F(X^{n_A}_{-\ell_1,-\ell_2})^2 \cdot \frac{1}{F(X^{n_A}_{-\ell_1,-\ell_2})^2} = 1. \tag{19}$$

Since $E(V_{\ell_1} V_{\ell_2} | X^{n_A}_{-\ell_1,-\ell_2}) = 1$ when conditioned on each $X^{n_A}_{-\ell_1,-\ell_2}$, we also have $E(V_{\ell_1} V_{\ell_2}) = 1$ when the expectation is taken over all X^{n_A}. Meanwhile, since T satisfies 1-Goodness, $E(V_{\ell_1}) = E(V_{\ell_2}) = 1$ (cf. Theorem 7). Hence, $\sigma(V_{\ell_1}, V_{\ell_2}) = 0$.

As a corollary of Lemma 3, we obtain the following result, establishing that the variance of $\hat{n}_{P,A}$ is equal to the sum of the variances of the per-identifier estimates for all identifiers in A satisfying property P.

Lemma 4. *Suppose that T satisfies* 1*-Goodness. Fix any stream A, and let $\hat{n}_{P,A}$ denote the estimate for $n_{P,A}$ obtained by running* samp$[T]()$ *on A and feeding the resulting theta-sketch into* EstimateOnSubPopulation*(). Then*

$$\sigma^2(\hat{n}_{P,A}) = \sum_{\ell \in A:\ P(\ell)=1} \sigma^2(V_\ell).$$

Proof. Note that $\hat{n}_{P,A} = \sum_{\ell \in A:\ P(\ell)=1} V_\ell$. The claim then follows from Lemma 3 combined with the fact that the variance of the sum of random variables equals the sum of the variances, provided that the variables appearing in the sum are uncorrelated.

G.4 Completing the Proof of Theorem 9

Proof. For every ℓ that appears in the concatenated stream A^*, and for all $X^{n_{A^*}}$, we define the "per-identifier estimate" $V_\ell(X^{n_{A^*}})$ as in Equation (5) with $A = A^*$, and relate it to the threshold $F_\ell(X^{n_{A^*}}_{-\ell})$ as in Equation (6), also with $A = A^*$. It is then straightforward to verify that

$$\sigma^2(V_\ell | X^{n_{A^*}}_{-\ell}) = 1/F_\ell(X^{n_{A^*}}_{-\ell}) - 1. \tag{20}$$

Let T' be the TCF that was (implicitly) used to construct (θ^U, S^U) from the m sketches of the individual streams A_j. By Theorem 6, T' satisfies 1-Goodness, so let $F'_\ell(X^{n_{A^*}}_{-\ell})$ denote the corresponding threshold value for T' as in Equation (6). We claim that T' satisfies the following property:

$$\text{For all identifiers } \ell \in [n] \text{ and for all } X^{n_{A^*}}, F'_\ell(X^{n_{A^*}}_{-\ell}) \geq F_\ell(X^{n_{A^*}}_{-\ell}). \tag{21}$$

Finishing the proof, assuming T' satisfies Property 21. By Equation (20):

$$\sigma^2(V'_\ell | X^{n_{A^*}}_{-\ell}) \leq \sigma^2(V_\ell | X^{n_{A^*}}_{-\ell}). \tag{22}$$

Because this inequality holds for every specific $X^{n_{A^*}}_{-\ell}$, it also holds for any convex combination over $X^{n_{A^*}}_{-\ell}$'s, so

$$\sigma^2(V'_\ell) \leq \sigma^2(V_\ell).$$

Combining this with Lemma 4, we conclude that

$$\sigma^2(\hat{n}^U_{P,U}) \;=\; \sum_{\ell \in A : P(\ell)=1} \sigma^2(V'_\ell) \;\leq\; \sum_{\ell \in A : P(\ell)=1} \sigma^2(V_\ell) \;=\; \sigma^2(\hat{n}^{A^*}_{P,A^*}).$$

Proving that T' satisfies Property 21. Fix any hash function h, which determines X^{n_U}, and also fixes hashed versions of the streams $A_1, \ldots, A_m$ and A^*. We will overload the symbols A_j and A^* to denote these hashed streams as well as the original streams. We need to prove that $F'_\ell(X^{n_U}_{-\ell}) \geq F_\ell(A^*_{-\ell})$. This can be done in three steps. First, from the proof of Theorem 6 we know that there exists a j such that $F'_\ell(X^{n_U}_{-\ell}) = F_\ell(A_{j,-\ell})$. Second, because T satisfies 1-Goodness, $F_\ell(A_{j,-\ell}) = T(Z(A_j, \ell))$ and $F_\ell(A^*_{-\ell}) = T(Z(A^*, \ell))$, where Z is a function that makes a copy of a hashed stream in which $h(\ell)$ has been artificially set to zero. Third, $Z(A^*, \ell))$ can be rewritten as the concatenation of 3 streams as follows: $B_0 \circ Z(A_j, \ell) \circ B_2$, where $B_0 = Z(A_1, \ell) \circ Z(A_2, \ell) \circ \ldots, Z(A_{j-1}, \ell)$, and $B_2 = Z(A_{j+1}, \ell) \circ \cdots \circ Z(A_m, \ell)$. Because T was assumed to satisfy the monotonicity condition, Condition 8, we then have

$$F'_\ell(X^{n_U}_{-\ell}) = T(Z(A_j, \ell)) \geq T(B_1 \circ Z(A_j, \ell) \circ B_3) = T(Z(A^*, \ell)) = F_\ell(A^*_{-\ell}). \tag{23}$$

H The pKMV Variant of KMV: Full Motivation and Analysis

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 Theta 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 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 7), T also satisfies 1-Goodness.

It is easy to see that Condition 8 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.

I The Alpha Algorithm

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. The Alpha Algorithm shares the advantages of KMV and Adaptive Sampling, relative to HLL. First, unlike HLL, the Alpha Algorithm provides unbiased estimates for $\textsc{Distinct}_P$ queries, for non-trivial predicates P. Second, 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 on multiple streams (cf. Section A.2). Third, because the Alpha Algorithm computes a sample of the stream, 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 [16] to give quick access to m_k (the k'th smallest 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 A.1). The biggest advantage of the Alpha Algorithm over KMV is that it avoids the need for a heap or Quickselect, yielding superior practical performance. In addition, the HIP estimator (cf. Appendix A) for the Alpha Algorithm applied to a single stream has a very simple closed form expression, and can be computed in constant time from the state of the sketch *after* the stream has been processed, with no additional work necessary during the stream processing phase.

Advantages over Adaptive Sampling. The accuracy of Adaptive Sampling oscillates as n increases. The Alpha Algorithm avoids this "saw-toothed" error behavior.

I.1 AlphaTCF

Algorithm 2 describes the threshold choosing function AlphaTCF that creates the instantiation of the Theta Sketch Framework whose base algorithm we refer to as the Alpha Algorithm. 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 Approximate Counting [18] 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, thus limiting the growth of that set.

If the set D is implemented using a standard dynamically-resized hash table, then well-known results imply that the amortized cost of processing each stream element is $O(1)$, and the space occupied by the hash table is $O(|D|)$.[11] For

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

Algorithm 2 The Alpha Algorithm's Threshold Choosing Function

```
Function AlphaTCF (target size $k$, stream $A$, hash function $h$)
$\alpha \leftarrow k/(k+1)$.
prefix($h(A)$) $\leftarrow$ shortest prefix of $h(A)$ containing exactly $k$ unique hash values.
suffix($h(A)$) $\leftarrow$ the corresponding suffix.
$D \leftarrow$ the set of unique hash values in prefix($H(A)$).
$i \leftarrow 0$.
for all $x \in$ suffix($h(A)$) do
  if $x < \alpha^i$ then
    if $x \notin D$ then
      $i \leftarrow i + 1$.
      $D \leftarrow D \cup \{x\}$.
    end if
  end if
end for
return $\theta \leftarrow \alpha^i$.
```

the algorithm as written in Algorithm 2, the expected size of the set D grows logarithmically with n, but an optimized implementation will periodically purge D by removing all items that are not in S. In Theorem 13 below, it is proved that $|S|$ is tightly concentrated around k, so this implementation's space usage is $O(k)$ in practice.

Section Roadmap. In the remainder of this section we will study the instantiation of the Theta Sketch Framework that is obtained by plugging in the threshold choosing function AlphaTCF. First, in Section I.2 we will prove the main result of this section, which is that AlphaTCF satisfies 1-Goodness, implying, via Theorem 7 that EstimateOnSubPopulation() is unbiased on single streams and on unions and intersections of streams in the framework instantiation created by plugging in AlphaTCF. Second, in Section I.3 we will prove that for theta-sketches (θ, S) created from a single stream A by samp[$AlphaTCF$], $|S|$ is tightly concentrated around k. Specifically: $E(|S|) = k$, and $\sigma^2(|S|) < \frac{k}{2} + \frac{1}{4}$. Third, in Section I.4 we will prove that for theta-sketches (θ, S) created from single streams by samp[$AlphaTCF$], $\sigma^2(|S|/\theta) < \frac{n_A^2}{k-\frac{1}{2}}$.

I.2 AlphaTCF Satisfies 1-Goodness

We will now prove that AlphaTCF satisfies 1-Goodness.

Theorem 12. *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.*

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 the the sequence of values $X^{n_A}_{-\ell}$. We need to exhibit a threshold F such that $x < F$ implies $T_\ell[X^{n_A}_{-\ell}](x_\ell)(x) = F$ and $x \geq F$ implies $T_\ell[X^{n_A}_{-\ell}](x) \leq x$.

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

```
                     b
              --------
      Rule 1 /
            /
           /
a ------- x
           \
            \
      Rule 0 \       c
              --------
```

Fig. 3. Visual depiction of Rules 0 and 1 appearing in the proof of Theorem 12

.

Now for the main case, suppose that ℓ does not lie in one of the first $k+1$ positions of the stream. Consider Figure I.2. We have subdivided 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 $X^{n_A}_{-\ell}$, 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
0	$x < \alpha^a$	α^{a+b+1}
1	$x \geq \alpha^a$	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 $T_\ell[X^{n_A}_{-\ell}](x) = F$. By Rule 0, $T_\ell[X^{n_A}_{-\ell}](x) = \alpha^{a+b+1} = F$.

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

Case 3: $F < \alpha^a \leq x$. Because $x \geq F$, we need to show that $T_\ell[X^{n_A}_{-\ell}](x) \leq x$. By Rule 2, $T_\ell[X^{n_A}_{-\ell}](x) = \alpha^{a+c+0} \leq \alpha^a \leq x$.

I.3 Analysis of the Size of the Set S

Let S be the set produced by Line 3 of Algorithm 1 when AlphaTCF is plugged into the Theta Sketch Framework, and let $\mathcal{S}$ be the random variable corresponding to $|S|$. In this section we compute $E(\mathcal{S})$ and bound $\sigma^2(\mathcal{S})$. We prove the top-level theorem using a lemma. The proofs of the theorem and lemma both involve two levels of conditioning. First we condition on the value of i when Line 15 of Algorithm 2 is reached. Then we further condition on J^+, which is the particular set of i stream positions on which increments occurred in Line 10 of Algorithm 2.

Theorem 13.

$$\mathrm{E}(\mathcal{S}) = k. \tag{24}$$

$$\sigma^2(\mathcal{S}) < \frac{k}{2} + \frac{1}{4}. \tag{25}$$

Proof. We will briefly summarize the argument for $\mathrm{E}(\mathcal{S})$. The argument for $\sigma^2(\mathcal{S})$ is only slightly more complicated. Details can be found in the extended version of the paper. First, we perform the following standard decomposition:

$$E(\mathcal{S}) = \sum_i \Pr(i) \sum_{J^+} \Pr(J^+|i) \underset{i,J^+}{\mathrm{E}}(\mathcal{S}|i, J^+) \tag{26}$$

Then, starting from Equation 27 in Lemma 5, we twice apply the fact that a convex combination of multiple copies of the same constant value equals that constant value.

Lemma 5.

$$\underset{i,J^+}{\mathrm{E}}(\mathcal{S}|i, J^+) = k, \tag{27}$$

$$\underset{i,J^+}{\mathrm{E}}(\mathcal{S}^2|i, J^+) < k^2 + \frac{k}{2} + \frac{1}{4}. \tag{28}$$

Proof. Let A be a uniquified stream of length n. Let h be a hash function that is chosen randomly. Let $\{X_p | 1 \leq p \leq n\}$ be a sequence of n *iid* random variables, one per stream position, each drawn from the distribution Uniform(0,1). Let X^{n_A} be the cross product of the X_p's; this random variable is our model of $h(A)$. Let I be a random variable generated by first choosing a random X^{n_A}, then running Algorithm 2 on X^{n_A}, and then setting I to be the value of the program variable i when Line 15 is reached. Define a set of n Bernoulli random variables S_p, one per stream position, derived from the variable I and the variables X_p by the rule $S_p = 1$ iff $X_p < \alpha^i$. The S_p's are *not* independent of each other, but become independent after conditioning on i.

Now we will describe the effect of conditioning on both i and J^+, by first describing how the original variables X_p are transformed into modified variables Y_p that are drawn from specific subintervals of $(0, 1)$. We will then introduce new Bernoulli variables S'_p defined by the rule $S'_p = 1$ iff $Y_p < \alpha^i$, and finally compute the expected value and variance of $(\mathcal{S}|i, J^+) = \sum_p S'_p$.

We are fixing a specific i and J^+; the latter is a size-i subset of the set of $n - k$ non-initial stream positions $\{k + 1, k + 2, \ldots, n - 1, n\}$. The set of $n - k - i$ non-initial stream positions that are not in J^+ will be referred to as

J^-. Let $f(p, J^+)$ be the function that maps any non-initial position p to the number of non-initial positions before p that are members of J^+. We note that for $p \in J^+$, $f(p, J^+) \in \{0, 1, \ldots, i-1\}$, and the mapping is one-to-one. For $p \in J^-$, $f(p, J^+) \in \{0, 1, \ldots, i\}$, and the mapping is not necessarily one-to-one.

Now we are ready to characterize the Y_p's and the S'_p's.

First let p be one of the k initial positions in the stream. In this case, conditioning on i and J^+ doesn't tell us anything about the value of X_p, so Y_p is drawn from the full interval $(0, 1)$, so $\Pr(Y_p < \alpha^i) = \alpha^i$; $E(\mathcal{S}'_p) = \alpha^i$, and $\sigma^2(S'_p) = \alpha^i(1 - \alpha^i)$.

Next, let p be one of the $n - k - i$ positions in J^-. For this position, the test in Line 8 of Algorithm 2 failed, so we know that $X_p \geq \alpha^{f(p,J^+)}$, so Y_p is drawn uniformly from the interval $[\alpha^{f(p,J^+)}, 1)$. Because $p \in J^-$, $f(p, J^+) \leq i$, so $\alpha^i \leq \alpha^{f(p,J^+)} \leq Y_p$, so $\Pr(Y_p < \alpha^i) = 0$, $E(S'_p) = 0$, and $\sigma^2(S'_p) = 0$.

Finally, let p be one of the i positions that are in J^+. For this position, the test in Line 8 of Algorithm 2 succeeded, so we know that $X_p < \alpha^{f(p,J^+)}$, so Y_p is drawn uniformly from the interval $(0, \alpha^{f(p,J^+)})$, so $\Pr(Y_p < \alpha^i) = \alpha^i/\alpha^{f(p,J^+)} = \alpha^{i-f(p,J^+)}$. Now, because $f(p, J^+)$ assumes each value in $\{0, 1, \ldots, i-1\}$ as p is varied over the contents of J^+, $E(S'_p) = \Pr(Y_p < \alpha^i) = \alpha^{i-f(p,J^+)}$ assumes each value in $\{\alpha^i, \alpha^{i-1}, \ldots, \alpha^1\}$. Similarly, $\sigma^2(S'_p)$ assumes each value in $\{\alpha^i(1-\alpha^i), \ldots \alpha^1(1-\alpha^1)\}$.

Putting together all of the above, and remembering that the random variables S'_p are independent due to the conditioning on i:

$$\begin{aligned}
E(\mathcal{S}|i, J^+) =& k \cdot \alpha^i + (n-k-i) \cdot 0 + \sum_{j=1}^{i} \alpha^j = k, \\
\sigma^2(\mathcal{S}|i, J^+) =& k\alpha^i \cdot (1-\alpha^i) + (n-k-i) \cdot 0 + \sum_{j=1}^{i} \alpha^j(1-\alpha^j) \\
=& \frac{\alpha - \alpha^{2i+1}}{1-\alpha^2}, \\
E(\mathcal{S}^2|i, J^+) =& k^2 + \frac{\alpha - \alpha^{2i+1}}{1-\alpha^2} < k^2 + \frac{k}{2} + \frac{1}{4}.
\end{aligned}$$

I.4 Variance of the Alpha Algorithm in the Single-Stream Setting

In Theorem 12 above, we proved that the Alpha Algorithm's threshold choosing function satisfies 1-Goodness, from which we know (via the results in Section 3) that the estimator of the Alpha Algorithm instantiation of the Theta Sketch Framework is unbiased on any subset of any stream or union of streams. However, we have not yet discussed the variance of this estimator. Theorem 14 provides such a bound on the Alpha algorithm's variance when applied to single streams. Theorem 9 implies that the same bound holds in the multi-stream setting, when instantiating the Theta-Sketch framework with AlphaTCF as the TCF.

Theorem 14.

$$\begin{aligned}\sigma^2(|S|/\theta) =& \frac{(2k+1)n_A^2 - (k^2+k)(2n_A-1) - n}{2k^2} \\ <& \frac{n_A^2}{k-\frac{1}{2}}.\end{aligned}$$

Proof. The proof combines our unbiasedness result for this estimator, a sub-result [now omitted] from our analysis of $|S|$, and a technical lemma concerning the distribution of $i(final)$ that can be proved using recurrences similar to those in [8]. Details appear in the extended version of this paper.

I.5 Variance of the Alpha Algorithm in the Multi-Stream Setting

Unfortunately, the Alpha Algorithm does not satisfies monotonicity (Condition 8) in general, and hence Theorem 9 does not immediately imply variance bounds for the 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 9 do apply to the Alpha Algorithm under the promise that $A_1, \ldots, A_m$ are pairwise disjoint. Our experiments in Section J suggest that, in practice, the variance of the Alpha Algorithm in the multi-stream setting is not much larger than in the pairwise disjoint case.

Theorem 15. *The TCF computed by the Alpha Algorithm satisfies Condition 8 under the promise that the streams A_1, A_2, A_3 appearing in Condition 8 are pairwise disjoint.*

Proof. Inspection of Algorithm 2 shows that the Alpha Algorithm never increases θ while processing a stream. Therefore, processing A_3 after A_2 cannot increase θ above the value that it had at the end of processing A_2. Hence, it will suffice to prove that $T(A_2) \geq T(A_1 \circ A_2)$. Referring to Line 15 of the pseudocode, we see that $\theta = \alpha^I$, where I is the final value of the program variable i, so it suffices to prove that $I(A_2) \leq I(A_1 \circ A_2)$.

We will compare two execution paths of the Alpha Algorithm. The first path results from processing A_2 by itself. The second path results from processing $A_1 \circ A_2$. We will now index the sequence of hash values of $h(A_1 \circ A_2)$ in a special way: x_0 will be the first hash value that reaches Line 8 of the pseudocode during the first execution path (where A_2 is processed by itself). Elements of $h(A_1 \circ A_2)$ that follow x_0 will be numbered $x_1, x_2, \ldots$, while elements of $h(A_1 \circ A_2)$ that precede x_0 will be numbered $\ldots, x_{-2}, x_{-1}$. To clarify, the boundary between negative and positive indices does not coincide with the boundary between A_1 and A_2.

For $j \geq 0$, let $I(j)$ denote the value of the program variable i immediately before processing the hash value x_j on the first execution path (A_2 alone), and let $I'(j)$ denote the same quantity for the second execution path ($A_1 \circ A_2$). We will prove by induction that for all $j \geq 0$, $I(j) \leq I'(j)$. The base case is trivial: by construction of our indexing scheme, at position 0, execution path one has had no opportunities yet to increment i, while execution path two might have had some opportunities to increment i. Hence $I(0) = 0$ while $I'(0) \geq 0$.

Now for the induction step. At position j, $I(j) \leq I'(j)$, and the two values of i are both integers, so the only possible way for $I(j+1) > I'(j+1)$ to occur would be for $I(j) = I'(j)$, and for the tests at Line 8 and Line 9 of the pseudocode to both *pass* on the first execution path, while at least one of them *fails* on the second execution path. However, the test in Line 8 must have the same outcome for both paths, since they are comparing the same hash value x_j against the same threshold $\alpha^i = \alpha^{i'}$. Also, given the assumption that A_1 and A_2 are disjoint, the "novelty test" in Line 9 is determined solely by novelty within A_2. Hence, it must have the same outcome on both paths. We conclude that it is impossible for i to be incremented on the first path but not on the second path, so $I(j+1) > I'(j+1)$ is impossible.

I.6 HIP estimator

For single streams, the HIP estimator (see Appendix A) derived from the Alpha Algorithm turns out to equal k/α^i. Notice that this estimator does not involve the size of the sample set S and is therefore not the same thing as the estimator $|S|/\alpha^i$ derived by instantiating the Theta-Sketch Framework with the Alpha Algorithm. The following theorem can be derived directly using the analysis of Approximate Counting that appears in [18] and [8].

Theorem 16. *If* $\alpha^i = \text{AlphaTCF}(k, A, h)$*, then*

$$\begin{aligned} \mathrm{E}(k/\alpha^i) =& n, \\ \sigma^2(k/\alpha^i) =& \frac{n^2 - 2nk + k^2 - n + k}{2k} < \frac{n^2}{2k}, \\ \text{S.E.}(k/\alpha^i) <& 0.708/\sqrt{k}. \end{aligned}$$

Note that the variance bound of the HIP estimator guaranteed by Theorem 16 is smaller than the variance bound for the vanilla Alpha Algorithm (cf. Theorem 14) by a factor of 2.

J Experiments

In this section we describe simulations showing that implementations of KMV, Adaptive Sampling, and the Alpha Algorithm can provide different tradeoffs between time, space, and accuracy. All three implementations take advantage of a version of cuckoo hashing that treats as empty all slots containing hash values that are not less than the current value of θ.

The code for our streaming implementation of the Alpha Algorithm closely resembles the pseudocode presented as Algorithm 2. The de-duping set D is stored in a cuckoo hash table that uses the just-mentioned self-cleaning trick. Hence D is in fact *always* equal to S, with no extra work needed in the form of table rebuilds or explicit delete operations.

Our implementation of Adaptive Sampling uses the same self-cleaning hash tables, but has a different rule for reducing θ: multiply by $1/2$ each time $|S|$ reaches a pre-specified limit. Again, no delete operations or table rebuilds are needed, but this program needs to scan the table after each reduction in θ to discover the current size of $|S|$.

Finally, our implementation of KMV again uses the same self-cleaning hash tables, but it also uses a heap to keep track of the current value of $\theta = m_{k+1}$. Hence it either uses more space than the other two algorithms, or it suffers from a reduction in accuracy due to sharing the space budget between the hash table and the heap. Also, it is slower than the other two algorithms because it performs heap operations in addition to hash table operations.

We now present some simulation results comparing the speed and accuracy of our implementations of the three algorithms under two different sets of experimental conditions.

First, we compare under "equal-k" conditions, in which all three algorithms aim for $|S| = t/2$, where $t = 2^{16}$ denotes the size of the hash table. Adaptive Sampling is configured to oscillate between roughly $|S| = (1/3)t$ and $|S| = (2/3)t$. We remark that KMV consumes more space than the other two algorithms under these conditions because of its heap.

Second, we compare under "equal-space" conditions reflective of a live streaming system that needs to limit the amount of memory consumed by each sketch data structure. Under these conditions, KMV is forced to devote half of its space budget to the heap, while both Adaptive Sampling and the Alpha Algorithm are free to employ parameters that cause their hash tables to run at occupancy levels well over $1/2$. In detail, for KMV $|S| = (2/5)t$, for the Alpha Algorithm $|S| = (4/5)t$, while Adaptive Sampling oscillates between roughly $|S| = (2/5)t$ and $|S| = (4/5)t$.

Simulation results are plotted in Figure 4. Two things are obvious. First, the heap-based implementation of KMV is much slower than the other two algorithms. Second, the error curves of Adaptive Sampling have a strongly oscillating shape that can be undesirable in practice.

Under the equal-k conditions, the error curves of KMV and the Alpha Algorithm are so similar that they cannot be distinguished from each other in the plot. However, under the equal space conditions, the Alpha Algorithm's ability to operate at a high, steady occupancy level (of the hash table) causes its error to be the lowest of the three algorithms. This high, steady occupancy level also causes the Alpha Algorithm to be slightly slower than Adaptive sampling under these conditions, even though the latter needs to re-scan the table periodically, while the Alpha Algorithm does not.

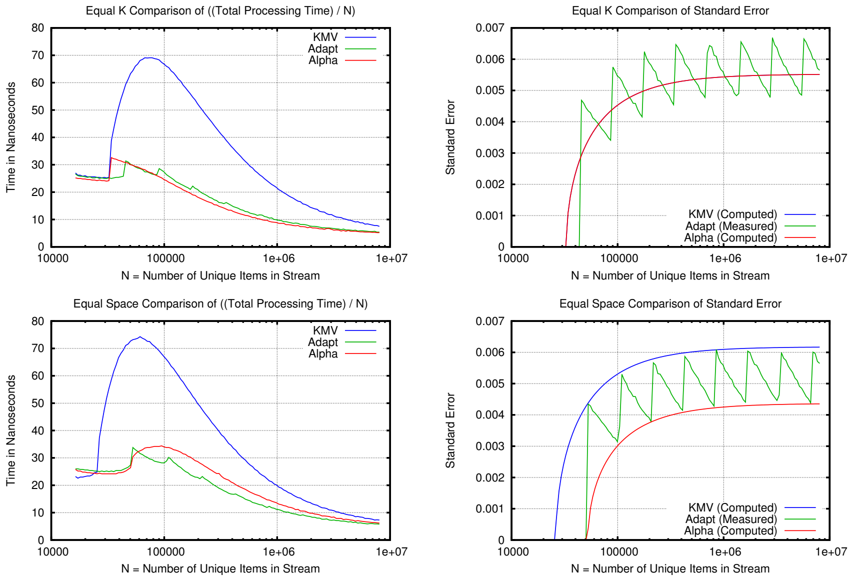


Fig. 4. These plots illustrate the low stream processing cost and smooth error curves of the Alpha Algorithm.

J.1 Variance of Alpha Algorithm in Multi-Stream Settings

As discussed in Section 3.7, Theorem 9's comparative variance result does not apply to the Alpha Algorithm in general. However, we proved in Appendix I that Theorem 9 does apply to the Alpha Algorithm when theinput streams are disjoint. In this section we present empirical evidence suggesting that the Alpha Algorithm "almost" satisfies the variance bound of Theorem 9 on real data. Recall that Theorem 9 asserted that $\sigma^2(\hat{n}^U_{P,U}) \leq \sigma^2(\hat{n}^{A^*}_{P,A*})$ when the estimates are computed using TCFs satisfying 1-Goodness and monotonicity. Simplifying notation, and switching from variance to relative error, we will exhibit a scatter plot comparing $\mathrm{RE}_U(A_1, A_2)$ versus $\mathrm{RE}_{A*}(A_1, A_2)$, for numerous pairs (A_1, A_2) of sets from a naturally occurring dataset, using the TCF defined by the Alpha Algorithm. This scatter plot will show that only a tiny fraction of the pairs violates the bound asserted in the theorem.

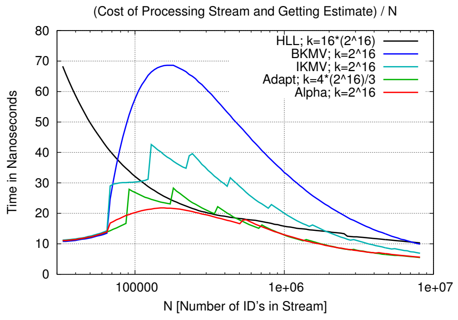


Fig. 5. Measured Stream Processing Cost

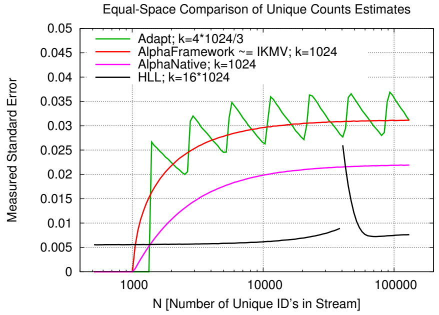


Fig. 6. Measured Accuracy for Single Streams

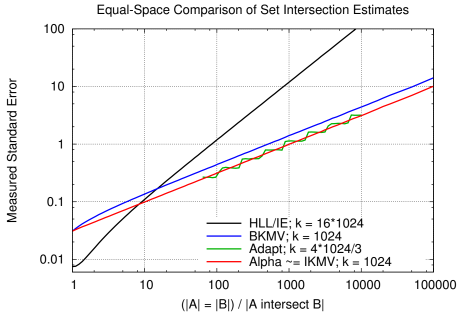


Fig. 7. Measured Accuracy for Set Intersections

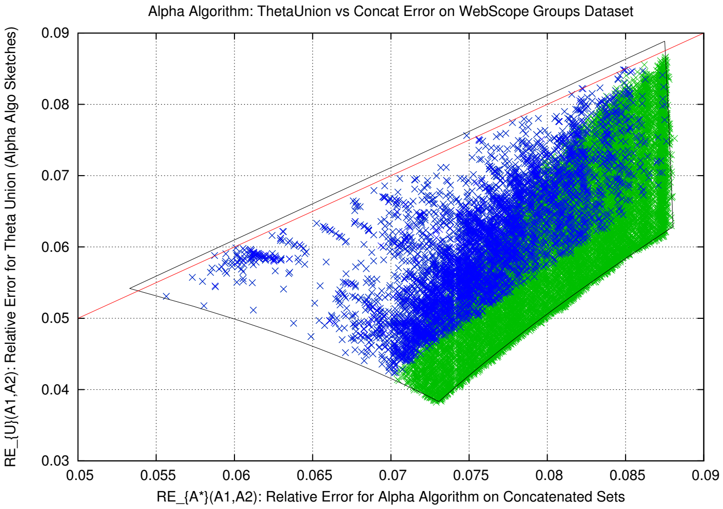


Fig. 8. Most points are below the red line, showing that the comparative variance bound of Theorem 9 is "nearly true" for the Alpha Algorithm on the Webscope Groups Dataset.

J.2 WebScope "Groups" Dataset

This experiment is based on ydata-ygroups-user-group-membership-graph-v1_0, a dataset that is available from the Yahoo Research Alliance Webscope program. It contains anonymized and downsampled membership lists for about 640000 Yahoo Groups, circa 2005. Because of the downsampling, there are only about 1 million members in all. We restricted our attention to the roughly 10000 groups whose membership lists contained between 201 and 5429 members. Hence there were about 50 million pairs of groups to consider. From these we examined all pairs with whose cosine similarity exceeded 0.3, because these high-overlap pairs seemed most likely to violate the theorem (as we have proved that the theorem applies to the Alpha Algorithm under the promise that the intersections are empty). There were about 5000 such pairs. We also examined another 13000 pairs to fill out the histogram.

For each of these roughly 18000 pairs of groups, we empirically measured, by means of 100000 trials with k set to 128, the values of $\mathrm{RE}_U(A_1, A_2)$ and $\mathrm{RE}_{A*}(A_1, A_2)$, and plotted them in the scatter plot appearing as Figure 8. The 5000 high-overlap pairs are plotted in blue, while the other 13000 pairs are plotted in green. Strikingly, all but 2 of the roughly 18000 points lie on or below below the red line, thus indicating an outcome that is consistent with the comparative variance result. Because we included every pair of sets that had large relative overlap (as measured by cosine similarity) we conjecture that all of the other roughly 50 million pairs of sets also conform to the theorem.

Figure 8 also includes a heuristic ‘bounding box” plotted as a black quadrilateral. This bounding box was computed numerically from several ingredients. For the $\mathrm{RE}_U(A_1, A_2)$ side of the computation, we exploited the fact that for any given values of n and k, the Alpha Algorithm's exact distribution over θ values can be computed by dynamic programming using recurrences similar to the ones described in [8]. To simplify the recurrences, we also made the (counter-factual) assumption that three different hash functions are used to process the input sets A_1 and A_2, and the output set S_U. This breaks the dependencies which make the multi-stream instantiation of the Alpha Algorithm, where there is only a single hash function during any given run. However, it also means that the resulting bounding box is not *quite* accurate. Finally, we did a grid search over all possible set-size pairs $201 \leq |A_1| \leq |A_2| \leq 5429$ and all possible amounts of overlap, and traced out the boundary of the resulting combinations of computed relative errors.

In addition to this evidence that the comparative variance result is true for nearly all pairs of sets that actually occurred in the data, the heuristic bounding box in Figure 8 suggests that it is true for nearly all *possible* triples $(|A_1|, |A_2|, |A_1 \cap A_2|)$ where $201 \leq |A_1| \leq |A_2| \leq 5429$ and $|A_1 \cap A_2| \leq A_1$. Moreover, in those relatively few cases where the theorem is violated, the magnitude of the violation is small.