Composable, Scalable, and Accurate Weight Summarization of Unaggregated Data Sets

Edith Cohen
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
edith@research.att.com

Nick Duffield
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
duffield@research.att.com

Haim Kaplan
School of Computer Science
Tel Aviv University
Tel Aviv, Israel
haimk@cs.tau.ac.il

Carsten Lund
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
lund@research.att.com

Mikkel Thorup
AT&T Labs–Research
180 Park Avenue
Florham Park, NJ 07932, USA
mthorup@research.att.com

ABSTRACT

Many data sets occur as *unaggregated data sets*, where multiple data points are associated with each key. In the *aggregate view* of the data, the *weight* of a key is the sum of the weights of data points associated with the key. Examples are measurements of IP packet header streams, distributed data streams produced by events registered by sensor networks, and Web page or multimedia requests to context distribution servers. We aim to combine sampling and aggregation to provide accurate and efficient summaries of the aggregate view. However, data points are scattered in time or across multiple servers and hence aggregation is subject to resource constraints on the size of summaries that can be stored or transmitted.

We develop a summarization framework for unaggregated data where summarization is a scalable and composable operator, and as such, can be tailored to meet resource constraints. Our summaries support unbiased estimates of the weight of subpopulations of keys specified using arbitrary selection predicates. While we prove that under such scenarios there is no variance optimal scheme, our estimators have the desirable properties that the variance is progressively closer to the minimum possible when applied to a "more" aggregated data set. An extensive evaluation using synthetic and real data sets shows that our summarization framework outperforms all existing schemes for this fundamental problem, even for the special and well-studied case of data streams.

1. INTRODUCTION

This paper concerns the problem of summarizing a large population of data points of the form (i, w) where i is a key and w a non-negative weight. The data are *unaggregated* in the sense that a key may appear in different data points. We aim to support queries on the total weight present over arbitrary selections of keys.

A motivating reference example comes from traffic measurement in computer communication networks, the data points being flow records that detail the number of bytes w that a router forwards between a source-destination pair i in some (usually short) period of time; typically multiple such data points would be produced by the router for each i over longer periods. We aim to find the total weight (bytes) of an arbitrary set of keys in an arbitrary time period.

It is important that we are not constrained to subsets known in advance of the measurements. This would preclude exploratory studies, and would not allow a change in routine questions to be applied retroactively to the measurements. A striking example where the selection is not known in advance was the tracing of the *Internet Slammer Worm* [20]. It turned out to have a simple signature in the flow record. Once this signature was identified, the worm could be studied by selecting records of flows matching this signature from the sampled flow records. We detail other applications in Section 2.

Our example is instructive in the sense that practical constraints for router design effectively rule out what might otherwise be considered an obvious application of a general approach, namely for the router to maintain an aggregate view of the data by summing the weights associated with each key. In a high speed communications network, too many distinct keys are present over any useful aggregation timescale to be concurrently stored and modified in a router memory that combines feasible cost, size and speed.

These constraints motivate a different general approach, which computes a *weight summary* of the population, comprising a random sample of keys, each of which has its associated weight adjusted so as to be an unbiased estimate of the exact aggregated weight for the key (see example in Figure 2). In keeping with our original aim, we can use the weight summary to estimate the weight of any key selection by summing the adjusted weights of those keys in the sample that are selected. As mentioned, the summary must be made without any prior knowledge of which key sets will be of interest for later selections. Finally, the summaries should minimize estimator variance.

1.1 Streaming for (Un)aggregated Data

Our reference example requires the summarization to work on data streams; the traffic measurement data points are generated by routers at an extremely high rate and hence the time window for buffering is extremely short. Beyond this, any measurement must either be immediately incorporated into the summary, or discarded forever. In fact, the problem of creating weight summaries has been studied intensely for streaming data. If all data points have distinct keys, then aggregation is trivial and the weight summaries are classic samples with associated estimates. For streams Knuth calls this reservoir sampling [19, pp. 138–140]: maintaining k samples for the part of the stream seen thus far; we review a variety of sampling strategies in Section 3.1. Whichever method is used, these samples should comprise adjusted weights for unbiased estimation. But the important general case of unaggregated data is much less understood. The previous techniques are most suited for the special case of streams where all point weights are unit. We shall review these techniques under related work in Section 3.2.

1.2 Information Flows on Trees

A further dimension to the problem of summarization that is present in some important applications is the hierarchical structure of measurement data flows. Again we use measurement in computer communications networks as a motivating example. In this case, hardware speed and cost constraints in the router typically prevent measurement on every packet flowing through the router. Instead routers perform summarization on a sampled substream of packets. Furthermore, bandwidth constraints in the measurement collection infrastructure prevent direct collection of the summaries from all routers. Instead, the collection topology is a tree, in which a set of co-located routers send their summaries to a local collector (e.g. one per city) that further summarizes data from all local routers, and forwards the summary to a regional collector, and so on, to a single final collector. We give more details applications, including the need for intermediate summarization, in Section 2.

We abstract this arrangement as an Information Flow Tree (IFT) in which information flows bottom up from the leaves to the root. Each node obtains weight summaries from its children, and summarizes these for its parent. Each node, including the root (the final collector) has a capacity constraint limiting the size of the summary it produces. Subjected to all the capacity constraints, we wish to obtain a summary that gives the most accurate weight estimates. Even though the weight summary produced by a node is aggregated with each key being distinct, a key may occur multiple times over the different summaries produced by it and its sibling nodes, hence the total input to the parent from all children is an unaggregated data set, albeit with random keys and weights.

The expressiveness of IFTs is illustrated in Figure 1. On the left we see how IFTs can capture the streaming model. The streaming IFT is a path with a single leaf hanging off each node. All edges have the same capacity, which corresponds to the storage limitation when processing the stream. An IFT for summarization of multiple distributed data streams over some communication network is illustrated in Figure 1 (middle). Edge capacities at each "stream" module capture storage constraints and other edge capacities capture network bandwidth constraints. Figure 1 (right) illustrates an IFT for summarization performed by multiple servers, each summarizing a part of the data and sending a summary of their part to a single central collector server, which produces a summary of the full data set. These IFTs also capture constraints of using multiple parallel processors: The data set is partitioned to processors, each processor produces a summary of its own chunk of the data, and these summaries are combined to produce the final output summary.

1.3 Contribution

So far we have established the aims and constraints for our work: we wish to summarize unaggregated data with unbiased estimator of arbitrary subset sums, having minimal variance, being computationally efficient, and respecting information flow constraints. The contribution of this paper is a summarization scheme which meets these goals; in particular:

- *Accuracy:* our new scheme is significantly more accurate than previous schemes both with heavy tailed weight distributions, and even in the special case of a stream of unit weights. We will document the improved accuracy experimentally on both real and synthetic data. Moreover, accuracy is sharp in the sense that estimator variance decreases for more aggregated input data, converging to variance-optimal sampling for completely aggregated input (i.e. having one data point per key).
- *Tree Modularity:* We frame our solution as a Summarization Algebra: each node in the IFT performs its own independent summarization of data below it in the tree, and this is the only information it passes to its parent. In principle, each node could use a different summarization algorithm.
- *Streaming Modularity:* Our scheme transparently allows streaming summarization. We use the following general approach to maintain a weight summary of k keys from *unaggregated* data. An incident data point is added to the adjusted weight if the key is included in the current summary. Otherwise, sampling is applied to decide which of the now $k+1$ keys to discard in order to keep within the limit k. We call this approach modular because any sampling algorithm can in principle be used to reduce from $k+1$ to k weights.
- *Multiple Weight Functions:* Our approach to unaggregated data is much more flexible than previous weight summarizations, and we can easily extend it to handle multiple weights, e.g., both byte and packet counts, and even negative weights.
- *Flexibility:* Our approach works naturally for non-streaming cases, e.g., distributed data collection by disjoint sensors that only have limited capacity for communication.
- *Efficiency:* We give a specialized implementation when each node uses the efficient variance optimal sampling scheme VarOpt from [6], that uses only $O(\log k)$ amortized CPU time per data point. The implementation is in practice much faster, being constant on non-pathological sequences. In experiments, reading an item from a file took about three times longer than summarizing it on a PC.

The layout of the remainder of the paper is as follows. Section 2 details further applications of our method. Section 3 discusses related work summarizing aggregated and unaggregated data, and the propagation of summaries on trees. Section 4 describes our modular approach to summarization, and previews the performance of our method relative to existing approaches. Section 5 describes adjusted weight summaries and their variance. Section 6 describes our proposed summarization algebra, while Section 7 describes its application under IFT constraints, and investigates the impact on estimator variance of unaggregated data. The scalable algorithms that facilitate our fast implementation are outlined in Section 8. Experimental comparison with other methods is in Section 9.

2. APPLICATIONS

As mentioned in the introduction, communication networks provide a fertile area for developing summarization methods. In the Internet Protocol (IP) suite, routers forward packets between high speed interfaces based on the contents of their packet headers. The header contains the source and destination address of the packets, and usually also source and destination port number which are used by end hosts to direct packets to the correct application executing

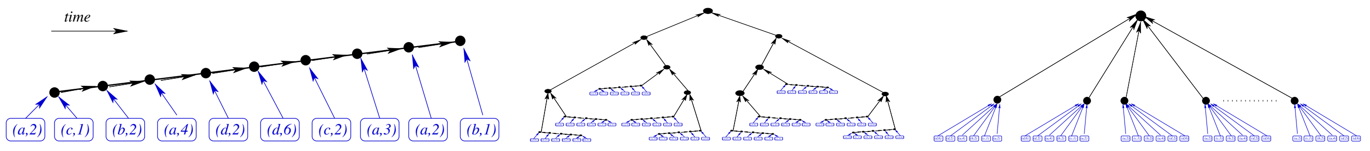

Figure 1: (left) IFT corresponding to data stream model constraints. The weighted set of this stream has keys $\{a, b, c, d\}$ with weights $w(a) = 2 + 4 + 3 = 9$. $w(b) = 2 + 1 = 3$, $w(c) = 1 + 2 = 4$, and $w(d) = 2 + 6 = 8$. The output of the root node is a summary of this weighted set. Other nodes output a summary of the respective prefix. For example, the third node outputs a summary of the weighted set with $w(a) = 6$, $w(b) = 2$, and $w(c) = 1$. (middle) IFT of an aggregation of multiple distributed data streams. (right)IFT for an aggregation of data from multiple servers sending summaries to a central server.

within them. These and other fields in the packet header constitute a key that identifies the IP flow that the packet belongs to. In our context, we can think of the set of keys of packets arriving at the router in some time interval, each paired with the byte size of the corresponding packet, as a population of unaggregated data points. As described in the introduction, routers typically sample packets and summarize the header information of a flows packets into flow records, including the key, aggregate packet and byte counts, along with timing and other information. Sampled NetFlow [22] is the most commonly deployed method in routers today. Other proposed methods for summarization are discussed in Section 3 below.

The flow records are exported up a measurement infrastructure, potentially involving multiple stages of summarization through sampling and aggregation, to a collector and ingested into a database. Common database queries for network administrators would include: (i) calculating the traffic matrix, i.e., the weight between source-destination address pairs; (ii) the application mix, as indicated by weight in various port numbers (iii) popular websites, as indicated by destination address using certain ports. Although some queries are routine, in exploratory and troubleshooting tasks the keys of interest are not known in advance.

A given flow record may serve different measurement applications, which operate with different levels of key granularity, and may require separate summarization within the measurement infrastructure. For example, service providers are commonly interested in the traffic matrix between different IP address routing prefixes, i.e., certain groups of IP addresses. Thus in a measurement module serving this applications, the original source-destination address-level key is mapped into a source-destination prefix-level key. Obviously, flow records having the same prefix-level key observed at a router can be aggregated, either by the router itself, or within the measurement infrastructure. But also at the prefix-level, with over 10^6 current prefixes, there are over 10^{12} possible keys and summarization is attractive for managing prefix-level aggregates. Whereas packets with shared address-level key typically follow the same path (same routers), and hence, the data source can be modeled as a stream, load balancing commonly causes different packets within the same prefix-level key to take different paths through the network, either on different parallel links between router pairs, or even traversing different routers in a city. Therefore, a suitable model this data source with prefix-level keys is distributed unaggregated streams.

Network devices that serve content or mediate network protocols generate logs comprising records of each transaction. Examples include web servers and caches; content distribution servers and caches; electronic libraries for software, video, music, books, papers; DNS and other protocol servers. We consider each record as a data point, keyed e.g. by requester or item requested, with weight being unity or the size or price of the item requested if appropriate. Offline libraries can produce similar records. Queries include finding the most popular items or the heaviest users, requiring aggregation over keys with common user and/or item. Another example is sensor networks that comprise of a distributed set of devices each of which generates monitoring events in certain categories.

3. RELATED WORK

3.1 Summarizing Aggregated Data

In aggregate data sets each data point has a unique key. There are many algorithms for computing a weight summary in reservoir/stream, offline, and distributed settings. These algorithms utilize uniform sampling [14, 32], Poisson sampling [16, 12] (with Horvitz Thompson [17] adjusted weights), probability proportional to size sampling with replacement (ppswr) [2], and order sampling [25, 8, 9] which includes pps without replacement [24] with adjusted weights in [9] and priority sampling [23, 11] with adjusted weights derived in [11].

Most of these methods are ill-suited to summarize unaggregated data. Pre-aggregation before sampling is generally prohibited by resource or streaming constraints[1]. Post-aggregation—after sampling effectively treating each data point as having a unique key—is inefficient for keys having large multiplicities of the same key and considerably less accurate than key-level summaries. This prompted the development of methods that compute key-level summaries over the unaggregated data.

3.1.1 Variance Optimal Summaries: VarOpt

VarOpt [6, 1, 30] is a generic weight summarization scheme for aggregated data that has optimality properties that date back to [28]. We leverage VarOpt in our modular summarization as the internal sampler at all nodes of an IFT.

For keys $[n] = \{1, ..., n\}$, let $w(i)$ denote the weight of key i and $\hat{w}(i)$ denote its corresponding adjusted weight in the summary. VarOpt$_k$ summary has the following properties:

(i) *Inclusion probabilities proportional to size (ipps).* Key i is included in the sample with probability $p(i) = \min\{1, w(i)/\tau_k\}$, where τ_k is the unique solution of $\sum_{i\in[n]} \min\{1, w(i)/\tau_k\} = k$ (assuming $k < n$; otherwise all keys are included with $p(i) = 1$).

The adjusted weight of a sampled key i is (the Horvitz-Thompson (HT) estimator [17]) $\hat{w}(i) = w(i)/p(i) = \max\{w(i), \tau_k\}$ (the weight divided by the probability that the key is included in the sample).

(ii) *Sample contains exactly* $\min\{k, n\}$ *keys.*

(iii) *No positive covariances* between distinct adjusted weights: $\textsc{cov}(\hat{w}(i), \hat{w}(j)) \leq 0$ for $i \neq j$.

[1] ASH and ANF [5, 3] realize the sample distribution of weighted sampling [24, 25], —see Section 3.2— but a second pass is required to determine weights.

(iv) *Total preserving:* $\sum_{i\in[n]} \hat{w}(i) = \sum_{i\in[n]} w(i)$.

HT adjusted weights are unbiased and minimize the variance for each key. In particular, they minimize $\Sigma V = \sum_{i\in[n]} \text{VAR}[\hat{w}(i)]$, for a given set of inclusion probabilities $p(i)$ $(i \in I)$. Ipps inclusion probabilities with HT adjusted weights minimize ΣV for a given average summary size [26, 27]. We discuss these qualities in more detail in Section 5.

3.2 Summarizing Unaggregated Data

Summarization of unaggregated data sets was studied for applications including data streams and stream databases [18], distributed data, and in-network aggregation (sensor networks). Our applications require estimation of arbitrary subset sums; we do not consider alternatives that do not, and hence exclude those restricted to estimate aggregates over full data, or limited aggregates such as top-k, heavy hitters, or frequency moments.

Concise samples [15] independently samples data points of uniform weight, aggregating all points with the same key, thus obtaining a larger effective sample using the same storage. This is also the flow counting mechanism deployed by Cisco's sampled NetFlow (NF) [22]. With fixed-rate sampling we obtain a variable-size summary. In many applications, a fixed-size summary is desirable, which is obtained by adaptively decreasing the sampling rate as new data points arrive. We refer to this adaptive version as ANF.

Counting samples [15] (see also sample-and-hold (SH) [13]) is a summarization algorithm applicable to an unaggregated data streams with uniform weights. It samples all data points at a fixed rate, but once a key is sampled, all subsequent data points with the same key are counted. Similarly, there is an adaptive version of the algorithm that produces fixed-size summaries (ASH) [15, 13].

Subpopulation-weight estimators for ASH and ANF were proposed and evaluated in [5, 3]. ASH is more accurate that ANF on any subpopulation and distribution. On the other hand NF and ANF are applicable on general IFTs, whereas ASH and SH are limited to streams. NF and ANF support multiple-objectives unbiased estimation for other additive (over data points) weight functions [4], whereas ASH and SH do not.

Step-counting SH (SSH) is another summarization scheme for unaggregated data streams that improves over ASH [5, 3] by exploiting the memory hierarchy of IP routers. As a pure data stream algorithm, however, SSH utilizes larger storage to produce the same size summary as ASH. Of the approaches considered so far, the best previous solutions we are aware of on data streams is ASH and on general IFT constraints, ANF.

3.3 Propagation of Summaries on Trees

Multistage aggregation for threshold sampling [12] is represented on a tree in [7] for the purpose of developing exponential bounds on estimation error. Applications include Sampled NetFlow, Counting Samples, and Sample and Hold. Some earlier work [10] had analyzed variance for Sampled NetFlow exploiting relationships similar to Lemma 6.8 for multistage sampling. However, both these cases were restricted to the independent sampling.

4. MODULAR SUMMARIZATION

In the modular summarization algebra (SA) outlined in Section 1.3, each node in an IFT passes up to its parent a further summaries of the weight summaries produced by its children. This internal summarization is performed by a sampling scheme $\mathcal{S}$ for *aggregated* data. If the same local scheme is used at all nodes, we denote the resulting scheme $\text{SA}[\mathcal{S}]$. In cases where the structure is specialized to some class $\mathcal{I}$ of IFTs we indicate this as an argument ($\mathcal{I}$): For streams (Figure 1 (left)), we use $\text{SA}[\mathcal{S}](\text{stream})$. For the 2-level model of Figure 1, we use $\text{SA}[\mathcal{S}](\text{servers})$. Our notation expresses the underlying modularity, which allows arbitrary internal samplers. A natural choice for this sampler is VAROPT, the variance optimal scheme for aggregated data sets. It follows from a general recurrence in [6] that if all data points in the input to the IFT have unique keys, then SA[VAROPT] is a VAROPT scheme, as defined in Section 3.1.1.

We are left with some intriguing questions, whose answers we now preview.

- *How does* SA[VAROPT] *relates to* VAROPT*?, that is, what is the cost, in terms of estimation quality, of not aggregating recurrent keys prior to summarization ?*

  SA[VAROPT] does not fully inherit the theoretical variance optimality of VAROPT (Section 3.1.1). In particular, inclusion probabilities are not ipps, adjusted weights are not Horvitz-Thompson, and hence ΣV is not minimized (property (i)). We show, however, that even in the streaming model, there does not exist a summarization scheme for unaggregated data that minimizes ΣV (see Section 7.2). On the flip side, SA[VAROPT] does have properties (ii)-(iv): fixed-size sample, unbiased estimation, and is total preserving with no positive covariances.

  We empirically compared SA[VAROPT] to the ideal optimal scheme which first performs complete aggregation of the data and then applies VAROPT. We found that on real data sets, SA[VAROPT] was never more than 15% from this unattainable optimum, and in many experiments, the difference was only a fraction of a percent.

- *How does* SA[VAROPT] *compare with existing schemes for unaggregated data?* In our experiments for data streams, our scheme SA[VAROPT](stream) typically had a 10%-40% reduction in variance over the more specialized ASH. For more general IFTs with distributed data or applications with multiple weight functions (see Section 7.3), the only competitor to our framework is ANF. We observed 3-10 fold reduction in variance when using SA[VAROPT] instead of ANF.

- *Is there an efficient scalable implementation of* SA[VAROPT]*?* We present a specialized implementation of SA[VAROPT](stream) that uses only $O(\log k)$ amortized CPU time per data point. The implementation is in practice much faster, being constant on non-pathological sequences.

5. WEIGHT SUMMARIES

Weighted Sets and Summaries. We explore in more detail the notion of a weight summary. We start with some universe U of possible keys. A weighted set is specified by a pair (I, w) where $I \subset U$ and w is a function from U to the non-negative real numbers, with the convention that w extends to 0 throughout $U \setminus I$. When U is clear from context, we sometimes specify a weighted set using the function w. Data points are singleton weighted sets of the form $(i, w(i))$.

DEFINITION 5.1. *A weight summary (I, A) of a weighted set (I, w) is a random weighted set such that for any key $i \in U$, $\mathsf{E}[A(i)] = w(i)$.*

The weights $A(i)$ of a weight summary are called adjusted weights. By linearity of expectation, the weighted sum $w(J) = \sum_{j\in J} w(j)$ over selected keys $J \subset U$ has the unbiased estimator $A(J) = \sum_{j\in J} A(j)$. $A(j)$ is computed over the summary, by first applying the selection predicate to included keys and adding

unaggregated data set with keys $I = \{a, b, c\}$:
$\{(a, 1), (b, 1), (a, 2), (c, 1), (b, 1)\}$
aggregated weights:
$w(a) = 3, w(b) = 2, w(c) = 1$
weighted set:
$(I, w) = \{(a, 3), (b, 2), (c, 1)\}$

weight summary A_1 of (I, w):

p	weights
1/2	$\{(a, 6)\}$
1/3	$\{(b, 6)\}$
1/6	$\{(c, 6)\}$

weight summary A_2 of (I, w):

p	weights
2/3	$\{(a, 3), (b, 3)\}$
1/3	$\{(a, 3), (c, 3)\}$

weight summary A_3 of (I, w):

p	weights
3/5	$\{(a, 5), (c, 1)\}$
2/5	$\{(b, 5), (c, 1)\}$

Figure 2: Example showing an unaggregated data set, the corresponding aggregated data as a weighted set (I, w), and three weight summaries distributions A_1, A_2, A_3 of (I, w). Keys with zero adjusted weights are not listed.

up the adjusted weights of keys that satisfy the predicate. Consider $J = \{a, b\}$ for the example in Figure 2 and estimating $w(J) = w(a) + w(b) = 3 + 2 = 5$ using A_2. With probability $2/3$ the summary includes keys a, b, which are both members of J. We obtain the estimate $A_2(J) = A_2(a) + A_2(b) = 3 + 3 = 6$. With probability $1/3$ the summary includes a, c. Only a is a member of J and hence $A_2(J) = A_2(a) = 3$. It is easy to see that the expectation $\mathsf{E}[A_2(J)] = 6 * (2/3) + 3 * (1/3) = 5 = w(J)$.

Summary Size. Different weight summaries of the same weighted set are compared based on their *size* and *estimation quality* tradeoffs. The size of a summary is the number of keys with positive adjusted weights. The *average size* of a weight summary is $\mathsf{E}[|\{i|A(i) > 0\}|]$. A weight summary has a *fixed size k* if it assigns positive adjusted weight to *exactly k* keys. The weight summaries in Figure 2 have fixed sizes. A_1 has size 1 and A_2 and A_3 have size 2.

Estimation quality measures. Variance is the standard metric for the quality of an estimator. The variance $\text{VAR}[A(J)]$ for a particular subpopulation J is equal to

$$\sum_{i,j \in J} \text{COV}_A[i, j] = \sum_{i \in J} \text{VAR}_A[i] + \sum_{i \neq j, i,j \in J} \text{COV}_A[i, j],$$

where $\text{COV}_A[i, j] \equiv \text{COV}[A(i), A(j)] = \mathsf{E}[A(i)A(j)] - w(i)w(j)$ is the covariance of $A(i)$ and $A(j)$. No single weight summary can dominate all others of the same size on all subpopulations. Since we are aiming for a summary that can be used with arbitrary subpopulations, the notion of a quality metric is more subtle.

The *average variance* of weight summary A of (I, w) over subpopulations of a certain cardinality is a linear combination of two quantities: the *sum of per-key variances* $\Sigma V[A] \equiv \sum_{i \in I} \text{VAR}_A[i]$ and the *variance of the sum* $V\Sigma[A] = \text{VAR}[A(I)]$ [29]. For A that *preserves total weight* ($A(I) = w(I)$, $V\Sigma[A] = 0$), the average variance is minimized when $\Sigma V[A]$ is minimized. The three weight summaries in Figure 2 are total preserving. We have $\text{VAR}_{A_2}[a] = 0$, $\text{VAR}_{A_2}[b] = \text{VAR}_{A_2}[c] = 2$, and hence $\Sigma V[A_2] = 4$. We have $\text{VAR}_{A_3}[a] = \text{VAR}_{A_3}[b] = 6$, $\text{VAR}_{A_3}[c] = 0$, and hence $\Sigma V[A_3] = 12$.

In practice, average variance bounds are insufficient as they do not translate into bounds on the estimation error of specific queries. This metric is complemented by limiting the covariance structure: A weight summary A has *non-positive covariances* if for every two keys $i \neq j$, $\text{COV}_A[i, j] \leq 0$. It is easy to see that the weight summariesin Figure 2 have non-positive covariances – for keys a, b in A_2 we have $\mathsf{E}[A_2(a)A_2(b)] = 9(2/3) = 6 \leq w(a)w(b) = 6$, for keys b, c we have $\mathsf{E}[A_2(b)A_2(c)] = 0 \leq w(b)w(c) = 2$.

The weight summarizations we propose and existing ones we evaluate (VarOpt, aNF, aSH) all preserve total weight, have fixed size, and have non-positive covariances. Therefore, all same-size sumamries can be compared using the sum of per-key variances ΣV. In Figure 2, both $\Sigma V[A_2]$ and $\Sigma V[A_3]$ have size 2 but $\Sigma V[A_2] < \Sigma V[A_3]$ and therefore A_2 is a better summary.

The three weight summaries in Figure 2 have HT adjusted weights, as each key obtains the same adjusted weight when it is included in the summary. The inclusion probabilities of (a, b, c) in A_1 are $(1/2, 1/3, 1/6)$ and in A_2 are $(1, 2/3, 1/3)$, and are proportional to their weights $(3, 2, 1)$. Therefore, A_1 and A_2 realize ipps sampling for size-1 and size-2 samples, which minimizes ΣV. Since they also have the other desirable properties, A_1 and A_2 are optimal (size-1 and size-2) VarOpt weight summaries of (I, w) [6, 1, 30] (see Section 3.1.1).

6. SUMMARIZATION ALGEBRA

Here we collect the specifics of the summarization algebra in the form of formal statements of the algebraic properties; for space reasons we omit the proofs. The *sum* of two weighted sets is defined by key wise addition over the union of keys, i.e., $(I_1, w_1) \oplus (I_2, w_2) = (I_1 \cup I_2, w_1 \oplus w_2)$ where $(w_1 \oplus w_2)(i) = w_1(i) + w_2(i)$ $(i \in U)$. Note this extends to the sum of multiple weighted sets $w_1 \oplus w_2 \oplus \cdots \oplus w_h = \bigoplus_{j=1}^{h} w_j$. Observe that the sum operation is commutative.

Some important properties, including being a weight summary, are *additive*. Let w_j $(1 \leq j \leq h)$ be weighted sets with respective weight summaries A_j. Let $w = \bigoplus_{j=1}^{h} w_j$.

Lemma 6.1. *The random weighted set $A = \bigoplus_{j=1}^{h} A_j$ is a weight summary of w.*

Clearly, if all A_j preserve the total weight, then so does $\bigoplus_{j=1}^{h} A_j$:

$$\sum_{i \in U} (\bigoplus_{j=1}^{h} A_j)(i) = \sum_{j=1}^{h} \sum_{i \in U} A_j(i) = \sum_{j=1}^{h} w_j(U) = w(U) .$$

Lemma 6.2. *If the weight summaries A_j are independent, then covariances are additive:* $\text{COV}_{\bigoplus_{\ell=1}^{h} A_\ell}[i, j] = \sum_{\ell=1}^{h} \text{COV}_{A_\ell}[i, j]$.

The sum of weight summaries preserves the non-positive covariances and zero covariances properties:

Corollary 6.3. *If the weight summaries A_j are independent, then if A_j $(j = 1, \ldots, h)$ have the non-positive covariances or the zero covariances properties then so does the weight summary $\bigoplus_{j=1}^{h} A_j$.*

Corollary 6.4. *If the weight summaries A_j are independent, then for each key $i \in U$,* $\text{VAR}_{\bigoplus_{j=1}^{h} A_j}[i] = \sum_{j=1}^{h} \text{VAR}_{A_j}[i]$.

Corollary 6.5. *If the weight summaries A_j are independent, then $\Sigma V[\bigoplus_{j=1}^{h} A_j] = \sum_{j=1}^{h} \Sigma V[A_j]$.*

Definition 6.6. *An adjusted-weights summarization scheme (weight summarization) $\mathcal{R}$ is a random mapping from the set of weighted sets into itself that for any weighted set w, $\mathcal{R}(w)$ is a weight summary of w.*

If we apply $\mathcal{R}$ to a random weighted set A, e.g. a weight summary, we use the notation $\mathcal{R} \circ A \equiv \mathcal{R}(A)$. It follows using conditional expectation and the law of total variance that weight summary properties are transitive under composition:

LEMMA 6.7. *Let A be a weight summary of w, and let $\mathcal{R}$ be a weight summarization defined over the support of A*

(i) $E[(\mathcal{R}(w))(i)] = w(i)$ *$(i \in U)$.*

(ii) $\mathcal{R} \circ A$ *is a weight summary of* w.

The composition $\mathcal{R}_1 \circ \mathcal{R}_2 \circ \cdots \circ \mathcal{R}_r$ (the domains must be compatible for this to be defined) of several weight summarizations is also a weight summarization. Suppose A, B are weight summaries of w with the property that $\mathsf{E}[B(i)|A] = A(i)$ for all $i \in U$. Then $\text{COV}_B[i,j|A]$ will denote the conditional covariance of $B(i), B(j)$, i.e., conditioned on A. Set $\text{VAR}_B[i|A] = \text{COV}_B[i,i|A]$. The following is a Law of Total (co)Variance for the present model.

LEMMA 6.8. *For each pair of keys $i, j \in U$,*

$$\text{COV}_{\mathcal{R}\circ A}[i,j] = E[\text{COV}_{\mathcal{R}\circ A}[i,j|A]] + \text{COV}_A[i,j].$$

COROLLARY 6.9. *For each key $i \in U$,*

$$\text{VAR}_{\mathcal{R}\circ A}[i] = \text{VAR}_A[i] + E[\text{VAR}_{\mathcal{R}\circ A}[i|A]] .$$

In particular, we have, in an obvious notation, $\Sigma V[\mathcal{R} \circ A] = \Sigma V[A] + \mathsf{E}[\Sigma V[\mathcal{R} \circ A|A]]$.

LEMMA 6.10. *If the weight summaries A and the range of $\mathcal{R}$ preserve total weight, have the zero covariances, or have the non-positive covariances properties, then so does the weight summary $\mathcal{R} \circ A$.*

7. IFT-CONSTRAINED SUMMARIES

The additivity and transitivity properties of weight summaries from Section 6 guarantee that if each basic summarization step at and below a node utilizes a weight summarization that is total-preserving with non-positive covariances, then the node's output weight summary is also total preserving with non-positive covariances. Note that for this property to hold, the IFT structure does not have to be fixed. The next operation (and hence the tree topology above the node) can depend on the output, and a summarization can itself depend on its input data points. Generally, we can consider a family of recursive IFTs, which allow, for example, for different arrival orders of data points or for variable size streams.

This flexibility is important because an IFT node does not necessarily corresponds to a physical node. Constraints at a physical node can also be modeled by an IFT. For example, capacity constraints at a physical node may preclude efficient set merging of summaries obtained as feeds from its children – it might be necessary to partition the inputs across multiple process, the data stored in external memory, and at the extreme, internal memory only suffices to store the output summary size. In this case, we model the constraints of this physical node by a family of IFTs. E.g., if internal memory is only sufficient to store the output summary size then constraints are modeled by a family of IFTs corresponding to unaggregated stream.

The basic primitive of data stream summarization is a weight summarization that inputs a weighted set of size $k+1$ and outputs a weighted set of size k (removes a single key). Interestingly, any weight summary that produces a size k weight summary from a size $k+1$ weighted set using HT adjusted weights has the non-positive covariances property. Note this is important for secondary weights—see Section 7.3— for it implies that they have non-positive covariances.

LEMMA 7.1. *Consider a weight summarization that for an input weighted set of size $k+1$ produces summaries of fixed size k, (for inputs that are already of size k, it returns the input set) and uses the* HT *adjusted weights. This weight summarization has non-positive covariances.*

PROOF. Consider a weighted set (I, w) where $|I| = k+1$ and let $p(i)$ be the probability that i is included in the summary. We have $\sum_{i\in I} p(i) = k$. These probabilities uniquely define the distribution over summaries. The one key that is not included is selected with probabilities $1-p(i)$. (This is only correct for selecting k out of $k+1$). Otherwise, the summary distribution is not determined uniquely). Consider two keys $i \neq j$ and let $A(i)$ and $A(j)$ be the adjusted weights. Consider the expectation $\mathsf{E}[A(i)A(j)]$. We have $A(i)A(j) = 0$ with probability $2 - p(i) - p(j)$ (exactly one of the items i or j is not included). With probability $p(i) + p(j) - 1$, we have both items included with respective adjusted weights $A(i) = w(i)/p(i)$ and $A(j) = w(j)/p(j)$. Therefore $\mathsf{E}[A(i)A(j)] = w(i)w(j)\frac{p(i)+p(j)-1}{p(i)p(j)} \leq w(i)w(j)$. If $p(i) < 1$ and $p(j) < 1$, the inequality is strict. $\square$

It turns out that there is a *unique* such weight summarization that is also total-preserving and minimizes ΣV, which means it is optimal for this $k+1$ to k keys primitive. We will denote this base VAROPT scheme by B-VAROPT$_k$ [1, 30, 31, 6]). We specify B-VAROPT$_k$ below as Algorithm 1. The scheme B-VAROPT$_k$ inputs a weighted set of size $k+1$. It uses the ipps probabilities (see Section 3.1.1) $p(i) = \min\{1, w(i)/\tau\}$, where τ is the solution of $\sum_{i\in A} \min\{1, w(i)/\tau\} = k$. Key i is then dropped with probability $q(i) = 1 - p(i)$. We have $\sum_{i\in A} q(i) = 1$ (exactly one key is dropped). All remaining keys $j \in A \setminus \{i\}$ get adjusted weights $\hat{w}(j) = w(j)/p(j) = \max\{\tau, w(j)\}$.

Algorithm 1: B-VAROPT$_k(A)$ where $|A| = k+1$.

```
let $\tau$ be such that $\sum_{a\in A} \min\{1, \hat{w}(a)/\tau\} = k$
for $a \in A$ do $p(a) \leftarrow \min\{1, \hat{w}(a)/\tau\}$
generate uniformly random $r \in U(0,1)$
maximize $d$ such that $\sum\{(1-p(a))|a \in A, a < d\} \leq r$.
remove $d$ from $A$
for $a \in A$ do $\hat{w}(a) \leftarrow \max\{\tau, \hat{w}(a)\}$
```

This leads to a simple but inefficient implementation of SA[VAROPT$_k$](stream) in Algorithm 2.

Algorithm 2: SA[B-VAROPT$_k$](stream). Simple but inefficient implementation. Maintains summary A with adjusted weights $\hat{w}(a), a \in A$.

```
$A \leftarrow \emptyset$;
while new data point $(a, w)$ arrives do
    if $a \in A$ then $\hat{w}(a) \leftarrow \hat{w}(a) + w$
    else
        include $a$ in $A$
        $\hat{w}(a) \leftarrow w$
        if $|A| = k+1$ then B-VAROPT$_k(A)$
```

In Figure 3 we contrast an example application of SA[B-VAROPT$_2$](stream) on an unaggregated data stream, with the action of VAROPT$_2$ on the respective aggregated data set. The resulting weight summariesare different: while both are total preserving with non-positive covariances, that produced by SA[B-VAROPT$_2$](stream) does not have HT adjusted weights.

7.1 Working within IFT constraints

Internal summarization at an IFT nodes can be performed by merging the weighted sets collected from its children, then applying a weight summarization to the merged set to reduce it as needed to satisfy the capacity constraint on the edge directed to its parent. But is it always beneficial to merge the inputs? Below we justify the rule: *merge before summarizing*, meaning that it is beneficial to

VarOpt$_2$ on the aggregated stream: $\{(a,1),(b,2),(c,1)\}$.

VarOpt$_2$ weight summary:

p	weight assignment
1/2	{(a,2),(b,2)}
1/2	{(b,2), (c,2)}

SA[B-varopt$_2$](stream) on the unaggregated stream (4 data points): $(c,1),(b,1),(a,1),(b,1)$

weight summary after 2 data points:

p	weights
1	{(b,1),(c,1)}

weight summary after 3 data points:

p	weights
1/3	{(a,1.5),(b,1.5)}
1/3	{(b,1.5), (c,1.5)}
1/3	{(a,1.5), (c,1.5)}

weight summary after 4 data points:

p	weights
1/3	{(a,1.5),(b,2.5)}
1/3	{(b,2.5), (c,1.5)}
1/6	{(a,2), (c,2)}
1/12	{(a,2), (b,2)}
1/12	{(b,2), (c,2)}

Figure 3: Example: The weight summary distributions of SA[B-varopt$_2$](stream) on an unaggregated data stream and of VarOpt$_2$ on the respective aggregated data set.

merge data points before applying B-varopt. Intuitively, merging before summarizing extends the base optimality derived from B-varopt (minimal ΣV) from being per data-point to the batch of data-points obtained from the children. Formally, for a weighted set (J, A) (representing the current summary) and additional data points $(i_1, w_1), \ldots, (i_r, w_r)$

$$\Sigma V[\text{VarOpt}_k(J,A) \oplus \bigoplus_{j=1}^{r} \{(i_r, w_r)\}] \leq \quad (1)$$
$$\Sigma V[\text{B-varopt}_k(\cdots \text{B-varopt}_k((J,A) \oplus \{(i_1,w_1)\}) \ldots \oplus \{(i_r,w_r)\})]$$

We have not specified a particular VarOpt$_k$ for input size larger than $k+1$; in this context we use B-varopt$_{k'-1} \circ \cdots \circ$B-varopt$_k$, where $k' = |\{J \cup \{a_1, \ldots, a_r\}\}|$; see [6]. The left hand side of (1), by optimality of VarOpt$_k$, is the minimum ΣV for size-k weight summaries of the weighted set $(J,A) \oplus \bigoplus_{j=1}^{r} \{(i_r, w_r)\}$. The right hand side is another weight summary of this weighted set.

In the example of Figure 3, $\Sigma V = 2$ by VarOpt$_2$, which is strictly lower than $\Sigma V = 2\frac{1}{3}$ by SA[B-varopt$_2$](stream) on the unaggregated data set. This is an example where the inequality (1 is strict and hence, the "merge before summarizing" rule results in a strictly better weight summary. This sharply contrasts aggregated data sets where there is no benefit to a VarOpt elimination procedure initiated with the full data set [30] over a reservoir implementation of VarOpt [1, 6].

If the data happens to be aggregated, and all intermediate summarizations allow for summary size that is at least the output size k, then SA[VarOpt] is an instance of VarOpt$_k$.

7.2 The Cost of Not Aggregating

SA[VarOpt] on unaggregated data set is outperformed by VarOpt applied to a respective aggregated data set. SA[VarOpt] has the advantage that by leveraging B-varopt as a building block, ΣV gracefully converges to the optimal when the data is more aggregated and attains it if the data set happens to be aggregated. We show that this cost of not aggregating is inherent. That is, there is no IFT-constrained summarization algorithm of unaggregated data sets that (is guaranteed to) minimize ΣV.

Theorem 7.2. *There is no weight summarization algorithm for unaggregated streams that produces a fixed-size summary that minimizes* ΣV.

Proof. We find a weighted set (data below an IFT node) such that for *any* weight summary A of this set, there *exists* a "completion" into a *dominating* (has at least the weight for any key) weighted set (below a parent node) such that A is not the right building block of an optimal summary for the parent. Consider two streams that share the prefix $P = ((i_1,1),(i_2,1),(i_3,1))$. S_1 has P followed by $(i_3,1)$ and S_2 has P followed by $(i_4,1)$. Let $k=2$. We show that any weight summary of P, can not be completed to an optimal weight summary of both S_1 and S_2.

The minimum ΣV over size-2 weight summaries of a weighted set with positive weights $(1,1,2)$ (aggregating S_1 respectively for keys (i_1,i_2,i_3)) is 2. The unique size-2 weight summary with $\Sigma V = 2$ is $(0,2,2)$ with probability $1/2$ and $(2,0,2)$ with probability $1/2$.

The minimum ΣV over size-2 weight summaries of a set with positive weights $(1,1,1,1)$ (aggregated S_2) is 4, and is attained by assigning two keys adjusted weight 0 and the other two keys adjusted weight 2, with all keys having the same inclusion probability.

Consider two weight summaries, A_1 and A_2 for the prefix P. A_2 is (the unique) optimum for P and has probability $1/3$ for each of $(3/2,3/2,0)$ $(3/2,0,3/2)$ and $(0,3/2,3/2)$. A_1 has $(0,2,1)$ and $(2,0,1)$, each with probability $1/2$. We have $\Sigma V[\text{VarOpt}_2 \circ (A_1 \oplus (i_3,1))] = 2$. In fact A_1 is the unique size-2 weight summary of P that can be completed to a weight summary of S_1 with minimum ΣV. For any other weight summary, including A_2, we have $\Sigma V[\text{VarOpt}_2 \circ (A_2 \oplus (i_3,1))] > 2$. Similarly, $\Sigma V[\text{VarOpt}_2 \circ (A_2 \oplus (i_4,1))] = 4$ and A_2 is the unique size-2 weight summary of P with this property, for any other weight summary, including A_1, we have $\Sigma V[\text{VarOpt}_2 \circ (A_1 \oplus (i_4,1))] > 4$. □

A consequence of Theorem 7.2 is that *there is no optimal algorithm for unaggregated streams*, that is, an algorithm that outperforms all others on all inputs. In particular, there are examples where SA[VarOpt](stream) has a slightly larger ΣV than ASH. On the other hand, we constructed a family of unaggregated streams where ASH has larger ΣV by a *logarithmic* (in k) factor.

We conclude this theoretical discussion with a conjecture. We define the *competitive ratio* of a weight summarization A of size k as the worst-case ratio (over all applicable input data sets) between $\Sigma V[A]$ and the minimum possible ΣV ($\Sigma V[\text{VarOpt}_k]$ on the corresponding aggregated data). For SA[VarOpt] summary A of a data set and corresponding family of IFTs, we define k' to be the smallest size of an intermediate summary on which VarOpt is locally applied and consider the ratio $\Sigma V[A]/\Sigma V[\text{VarOpt}_{k'}]$. We conjecture that this ratio, and in particular, the competitive ratio of SA[VarOpt$_k$](stream) are bounded by a constant.

The competitive ratio of ASH is at least $\log k$ whereas the worst example we could find for SA[VarOpt$_k$](stream) (on a contrived family of sequences) was about 1.6. The highest ratio we observed on real-life data was at most 1.15 (see Section 9).

7.3 Multiple Objectives Estimates

Applications frequently require aggregate values with respect to multiple weight functions, as is the case with IP flow records with packet and byte counts. Each data point is modeled as a tuple (i, w, h), where i is a key, w is the primary weight and h is a secondary weight. (There can be multiple secondary weights $h_1, h_2, \cdots$, but they are all handled similarly.) As with primary weights, the secondary weight of a key is defined to be the sum of the secondary weights of data points that share the key.

We show that if nonzero secondary weight implies positive primary weight *pointwise*, our summarization algebra extends to support unbiased estimation over secondary weights. Summarization is performed with respect to the primary weights, but we maintain adjusted secondary weights $H(i)$ for each key in the summary. Adjusted secondary weights are updated as follows after summarization and addition operations: Consider a weight summary A' applied to a weighted set with primary weights A and secondary weights H. That is, for a key i, $A(i)$ and $H(i)$ are the weights of i before summarization and (the random variable) $A'(i)$ is the adjusted primary weight after summarization. We argue that $H'(i) = H(i)\frac{A'(i)}{A(i)}$ are correct adjusted secondary weights. Note that $H'(i)$ are well defined because we assumed that $H(i)$ is nonzero only if $A(i) > 0$. It is easy to see that $\mathsf{E}[H'(i)] = \mathsf{E}[A'(i)]H(i)/A(i) = H(i)$. Clearly, if $H'(i) \neq 0$ then $A'(i) \neq 0$. Hence, the final weight summary satisfies our assumption that nonzero secondary weight implies positive primary weights pointwise.

When summing two weighted sets, we perform a key wise addition of adjusted secondary weights (in addition to adjusted primary weights). This operation retains unbiasdness of adjusted weights. If each weighted set has the property that when the secondary weight is nonzero, the primary weight is positive, this property also holds for the result of the addition (using nonnegativity of primary weights).

If the weight summarization has non-positive covariances of primary weights, this property carries over to secondary weights: We substitute our assumption that $\mathsf{E}[A'(i)A'(j)] \leq A(i)A(j)$ in $\mathsf{E}[H'(i)H'(j)] = \frac{H(i)H(j)}{A(i)A(j)}\mathsf{E}[A'(i)A'(j)]$ and obtain $\mathsf{E}[H'(i)H'(j)] \leq H(i)H(j)$. On the flip side, adjusted secondary weights do not preserve total even if the primary weights do and they can have much higher variance compared to a summarization where they are considered primary weights, especially if the primary and secondary weights are weakly or negatively correlated.

Secondary weights are more flexible than primary weights in that they may assume negative values. Negative values are of interest because they naturally model deletions.

7.4 Signed weights

What if our primary weights are signed ? One solution is to use two summarization schemes, one for positive and one for negative data points. A more natural solution is to handle the weights as secondary but set and modify the (adjusted) primary weights as we please – as long as we maintain that the primary weights are positive when the secondary adjusted weight is nonzero, the adjusted secondary weights remain unbiased. We propose a heuristic where prior to each sampling step, each adjusted primary weight is resetted to the absolute value of respective adjusted secondary weight. If we use VarOpt for the sampling, then this heuristic guarantees that each sampling step locally minimizes ΣV.

7.5 Information flow DAGs

Our summarization framework can be extended to information flow DAGs where nodes can partition their output summary among multiple recipients. This extended model supports exporting selected keys out of a summary. This model is suitable when keys naturally expire, as is the case with IP flows that are considered final after certain time elapses since the arrival of the most recent packet. Expired records are final in the sense that all data points were already observed. These records are exported out of the fast-memory summarization module into a slower and larger storage module. This model also supports a continuous output of summarized records instead of periodic outputs of complete summaries. We leave details for future work.

8. IMPLEMENTATION

The naive implementation of SA[VarOpt] is inefficient. If the output weighted set of each application of B-VAROPT is transferred as a list, then each B-VAROPT$_k$ application performed after addition of data points requires $O(k)$ processing time. Similarly, without tuned data structures, the processing time of a merge (adding sets) depends on the sum of the sizes of the sets.

We designed and implemented an efficient code for SA[VarOpt] which maintains the summary in a tuned data structure that reduces worst-case per-data point processing to amortized $O(\log k)$. The implementation is fast and further benefits from the fact that the theoretical amortized $O(\log k)$ bound applies to worst-case distributions and arrangements of the data points. Performance on "real" sequences is closer to $O(1)$ time per data point that hold for randomly permuted data points.

Pseudocode for SA[VarOpt$_k$](stream)is provided in Algorithm 3. The algorithm input is an unaggregated stream of data points (a, w) where a is a key and w a positive weight. The summary A is a sample of up to k keys. Each included key a has an adjusted weight $\hat{w}(a)$. If a key is not in A its adjusted weight is 0.

Algorithm 3: SA[VarOpt$_k$](stream). Efficient implementation. Summary represented as $A = L \cup T$ with explicit adjusted weights $\hat{w}(a)$ if $a \in L$ while $\hat{w}(a) = \tau$ if $a \in T$.

```
$L \leftarrow \emptyset; T \leftarrow \emptyset; \tau \leftarrow 0$
while $|L| < k$ and new data point $(a, w)$ arrives do
    if $a \in L$ then $w(a) \leftarrow w(a) + w$
    else
        include $a$ in $L$
        $w(a) \leftarrow w$
if stream not ended then
    Insert $L$ in a heap maintaining the minimum key argmin$_{b \in L} w(b)$
    while new data point $(a, w)$ arrives do
        if $a \in A$ then
            if $a \in L$ then
                $w(a) \leftarrow w(a) + w$
            else
                $w(a) \leftarrow \tau + w$
                move $a$ from $T$ to $L$
        else
            $w(a) \leftarrow w$
            $X \leftarrow \emptyset$            /* keys to be moved from $L$ to $T$ */
            $small\_sum \leftarrow \tau * |T|$ /* sum of weights in $T \cup X$ */
            if $w > \tau$ then include $a$ in $L$
            else
                include $a$ in $X$
                $small\_sum \leftarrow small\_sum + w$
            while $small\_sum \geq (|T| + |X| - 1) \min_{b \in L} w(b)$ do
                $b \leftarrow$ argmin$_{b \in L} w(b)$
                move $b$ from $L$ to end of $X$
                $small\_sum \leftarrow small\_sum + w(b)$
            $t \leftarrow small\_sum/(|T| + |X| - 1)$
            Generate uniformly random $r \in U(0, 1)$
            if $r < |T|(1 - \tau/t)$ then
                $d \leftarrow \lfloor r/(1 - \tau/t) \rfloor$
                remove $T[d]$ from $T$
            else
                $r \leftarrow r - |T|(1 - \tau/t)$
                $d \leftarrow 0$
                while $r > 0$ do
                    $r \leftarrow r - (1 - w(X[d])/t)$
                    $d \leftarrow d + 1$
                remove $X[d]$ from $X$
            $\tau \leftarrow t$
            $T \leftarrow T \cup X$
```

When a new data point has been processed, we will have a summary represented by the following components:

- A threshold $\tau \in \mathbb{R}$.

The keys in A are partitioned into two sets T and L. Hash tables are used to determine membership of keys in A, T, and L.

- Each $a \in L$ has a weight $w(a) > \tau$. The set L is stored in a priority queue which always identifies the smallest weight $\min_{a \in L} w(a)$. Here, $\min \emptyset \equiv \infty$.
- Each $a \in T$ has a weight $w(a) \le \tau$. The set T is stored as a prefix of an array of size $k+1$.

For every $a \in A$, the adjusted weight is $\hat{w}(a) = \max\{\tau, w_a\}$. Thus $\hat{w}_a = w_a$ for $a \in L$ while $\hat{w}(a) = \tau$ for $a \in T$.

This design can be extended to support batch addition of data points into the summary (which requires allowing the size of the summary to vary.). When adding multiple summaries we maintain the internal representation of the largest child, and batch merge (and if necessary summarize) the input from other children. This tuning is important if all but one child have small weighted sets, as in the case of a stream (Algorithm 3) where one child represents the stream and the other a single data point to be added. To get $O(\log k)$ expected time per data point, and $O(1)$ for randomly permute streams, it is critical that we preserve our implicit representation.

9. EXPERIMENTAL EVALUATION

We evaluate our summarization scheme against best-known previous schemes on two basic IFT-constrained settings:

• **Unaggregated data streams.** We apply SA[VarOpt](stream) (our summarization algebra with B-VarOpt summarization in each step, see Algorithm 3). SA[VarOpt](stream) is compared with aSH (see Section 3).

• **Distributed servers** The unaggregated data set is partitioned to s *servers.* Each server produces a size-k summary of its data. These summaries are then sent to a "central collector" server, which adds up (fully merges) the summaries and produces a size-k summary of the entire data set (See Figure 1 (right) for the corresponding IFT constraints.) All servers apply VarOpt locally. We refer to this summarization as SA[VarOpt$_k$](servers). We compare SA[VarOpt$_k$](servers) with aNF (see Section 3). (aSH is not applicable in this model.)

aNF and aSH are defined for unit-weight data points, but their summaries depend only on the aggregated data set (Summaries are oblivious to IFT structure, stream order, or the level of aggregation of the data). For efficiency (eg, when summarizing bytes for IP flows), our implementation worked directly with the aggregated data. We used the best-known adjusted weighted sets based on aNF and aSH counts [3]: These weight summaries have non-positive covariances, are total preserving, and have no larger ΣV than all other respective methods.

Because all evaluated schemes are total preserving and have non-positive covariances, they can be compared using the ΣV metric (see discussion in Section 5). The relation of the ΣV values reflects the relation of the variances of the corresponding estimators on any subpopulation size. To demonstrate applications of usage for our summaries, we also evaluate the variance of the estimators on subpopulations defined by natural selection predicates.

As a reference point, we compute $\Sigma V[\text{VarOpt}_k]$ (exactly) on the *aggregated* data set. This is the minimum ΣV possible for the distribution and is not attainable for unaggregated data under the IFT constraints.

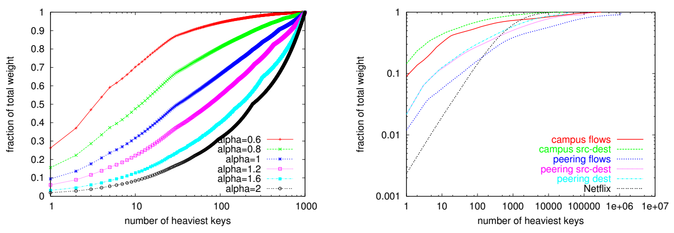

Figure 4: Left: Cumulative distributions of a Family of Pareto distributions on 1000 keys with parameters $\alpha = 0.6, 0.8, 1, 1.2, 1.6, 2$. Right: Netflix movies requests and IP flows bytes distributions.

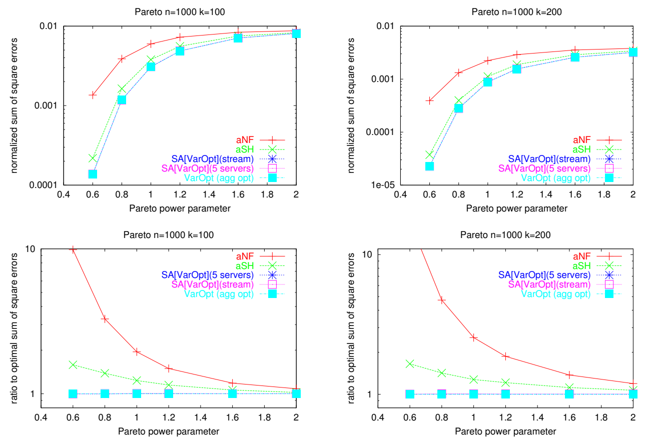

Figure 5: Pareto distributions, sweeping the power parameter α from 0.6 to 2 for a fixed sample size $k = 100$ (left) and $k = 200$ (right). Top: normalized average sum over keys of square errors. Bottom: the ratio of the sum of square errors to $\Sigma V[\text{VarOpt}_k]$.

9.1 Data sets

Pareto distributions. We generated a family of Pareto distributions, each with 1000 distinct keys, using different power parameters α (in the range 0.6 to 2). This synthetic data set enables us to explore how the relation between the different methods depends on the power α (skew of the distribution) and the summary size k. Properties of these distributions are illustrated in Figure 4(left). We can see that the *mean weight* of a key and the relative weight of the top-k set (the skew of the data) increase as α decreases.

We applied scaling and rounding to obtain integral weights with the smallest weight being 1 (to facilitate comparison with aSH and aNF that are only defined for integral weights). Corresponding unaggregated data sets were obtained by breaking each key into unit-weight data points.

Unaggregated data stream were obtained for each distribution by taking a random permutation of the data points. For the multiple servers model, we partition data points into 5 servers by randomly and independently assigning data points to servers.

Netflix Prize data. This data set [21] contains roughly 1×10^8 ratings of 17,770 distinct movies by roughly 5×10^5 users (anonymized user IDs). (See Figure 4(right)). We treat ratings as requests for movies from users.

Stream: We generate an unaggregated data stream from this data by treating all requests for a movie with the same time stamp (date) as a data point with weight equal to the number of requests and the

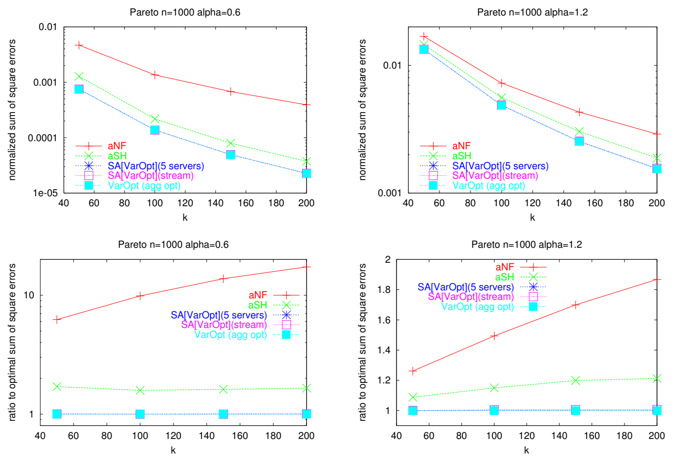

Figure 6: Pareto distributions, sweeping the sample size k for power parameter $\alpha = 0.6$ (left) and $\alpha = 1.2$ (right). Top: normalized average sum over keys of square errors. Bottom: the ratio of the sum of square errors to $\Sigma V[\text{VAROPT}_k]$.

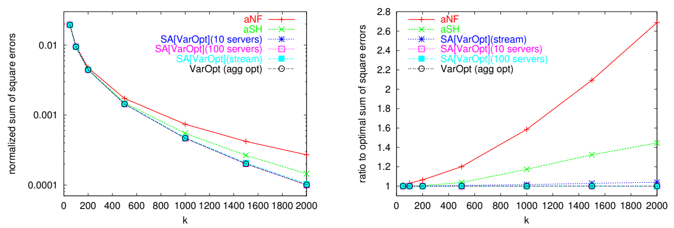

Figure 7: Netflix request stream, sweeping the sample size k. Left: normalized average sum of square errors. Right: the ratio of the sum of square errors to $\Sigma V[\text{VAROPT}_k]$.

movie ID as the key. These data points are naturally ordered by request date. The resulting stream has 1×10^7 data points.

Partition into "servers:" we obtained a natural partition the data into servers by mapping users to servers (all requests of the same user are mapped to the same server, selected uniformly at random). We used 10 and 100 servers.

Natural subpopulations: We evaluated the estimators on subpopulations defined by the release-year (and range of years) of the movie (release years range from 1896 to 2005).

IP packet header streams. We used two IP packet header streams gathered from network links, one in a campus network (*campus*) and another near a peering point (*peering*). The streams comprise metadata derived from each packet header, and the packet's weight in bytes. The analysis did not use raw header fields such as IP address. Instead, the data was anonymized prior to analysis, with header mapped to metadata comprising an application type derived from the TCP/UDP ports, and three key indices of different levels of granularity. The basic key level uses distinct values of the 4-tuple of source/destination IP address and TCP/UDP ports. Coarser keys used source and destination IP address, and destination IP address only. The key choice entails a tradeoff. While summarization based on coarser keys limits the selection predicate of the subpopulation, it allows for more accurate estimates using a given-size summaries for predicates based on the coarser keys. For example, a summary supporting destination-based selection (e.g., traffic to destinations in a certain AS or to news web servers) should use keys based only on destination IP address. If the summary needs to support more complex selections, based also on application and source IP, it should use the full flow attributes as the key.

The campus stream had 9.6×10^6 packets, 3.02×10^5 distinct flows, 4.7×10^9 bytes, 26×10^3 distinct source and destination IP addresses pairs, and 6.8×10^3 distinct destination IP addresses. The peering stream had 9.2×10^6 packets, 1.09×10^6 distinct flows, 4.25×10^9 bytes, 1.42×10^5 distinct source and destination IP addresses pairs, and 3.8×10^4 distinct destination IP addresses. (See Figure 4(right)).

Stream: The packet header stream is an unaggregated data stream. Each data point is a packet header metadata and the weight is the number of bytes.

Partition into "servers:" Typically all packets of a flow follow the same route. To obtain a meaningful partition, we used the peering data set with keys corresponding to destination IP address. Packets (data points) were assigned to servers by consistently assigning all the packets of the same (4-tuple) flow into the same random server. We used 100 servers. This way, each key (destination IP address) had data points in multiple servers (on average, each destination IP address had data points in about 10 servers) but all packets of a flow were assigned to the same server. This partition corresponds to load balancing of traffic across multiple routers. If each router produces a summary of destination addresses in the traffic it handles and these summaries are then combined into a global one of the full traffic.

Natural subpopulations: We considered subpopulations of flows based on application (identified by port number). From the peering stream, we selected three subpopulations that correspond to "web" (9.1×10^5 distinct flows, 2.45×10^9 total bytes) "p2p," (5.2×10^3 distinct flows, 3×10^8 total bytes) and "multimedia" (9.44×10^2 distinct flows, 9.49×10^7 total bytes) traffic.

From the peering stream with keys that correspond to destination IP addresses, we selected two (possibly overlapping) subpopulations of IP addresses: Destination IP addresses that support incoming web traffic (2.85×10^9 bytes and 2.79×10^4 distinct IP addresses) and destination IP addresses that support incoming p2p traffic (5.45×10^8 bytes and 1.165×10^3 addresses).

9.2 Results

We provide two plots that visualize the results for each data set. The first plot shows the (estimated) normalized ΣV of each method (sum over keys of square errors, averaged over at least 100 runs, normalized by dividing by the square of the total weight of the data set.) This plot is shown on a logarithmic scale to accommodate the wide range of ΣV as a function of summary size. Because the first plot uses a logarithmic scale, we include a second plot to illustrate the relation between the methods by showing the ratio of ΣV of each method to $\Sigma V[\text{VAROPT}_k]$. Results for both the stream and multiple servers models are shown on the same plots: SA[VAROPT](stream) should be compared with ASH and SA[VAROPT_k](servers) with ANF.

Figure 5 shows the results for Pareto distributions for a fixed summary size k and sweeping the power α. Figure 6 shows results for Pareto distributions with $\alpha = 0.6$ and $\alpha = 1.2$. Figure 7 shows results on the Netflix data. Figure 8 shows results for the two IP packet header streams with keys that correspond to IP flows. Figure 9 shows the results for the coarser keys (IP address pairs, and destination IP addresses) on the IP packet header streams.

For each selected subpopulation and weight summarization, we computed the average normalized square error of the estimate gleaned from the summaries (the square error is normalizing by dividing it with the square of the total weight of the subpopulation).

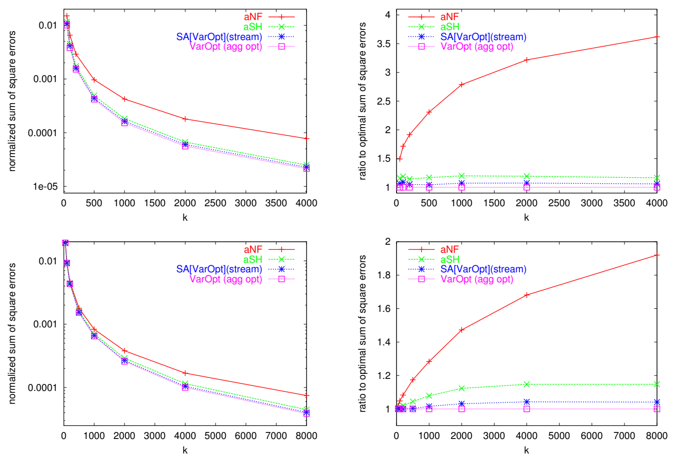

Figure 8: IP packet header streams with keys according to 4-tuple flows. campus (top) and peering (bottom). Left: normalized average sum of square errors. Right: the ratio of the average sum of square errors to $\Sigma V[\text{VarOpt}_k]$.

Note that the "offline" optimum is not meaningful for subpopulations (see discussion in Section 7.2) and therefore is not included in the plots. Figure 11 shows results for subpopulations of IP flows. Figure 10 (left and middle) shows results for subpopulations of destination IP addresses in the peering data set and Figure 10 (right) shows results for the Netflix data set.

We can see that across data sets, SA[VarOpt](stream) performs very closely to the (unattainable) optimum $\Sigma V[\text{VarOpt}_k]$. On Pareto data (randomly permuted packets), it is within fraction of a percent, within 5% on most datasets, and about 15% on one dataset. SA[VarOpt_k](servers) was closer to $\Sigma V[\text{VarOpt}_k]$ and within a fraction of a percent on all datasets. aNF is the worst performer, typically with 50% to an order of magnitude larger variance and aSH is in between. The Pareto distributions show that the performance gaps grow as the power parameter α decreases. Across data sets, performance gaps grow as k increases. Across data sets and subpopulations, on unaggregated data streams SA[VarOpt](stream) has $10\% - 40\%$ smaller variance than aSH. For multiple servers, SA[VarOpt_k](servers) has $2-10$ fold lower variance than aNF. Therefore, our summarization framework provides significantly tighter estimates.

10. CONCLUSION

This paper is motivated by the practical need for summarization through multilevel aggregation and sampling of high-rate, possibly distributed, streams of measurements of communications networks.

We develop a summarization algebra that reduces the summarization problem of unaggregated data when subjected to resource constraints to a primitive summarization operation on aggregated data sets. By using VarOpt as this primitive, a scheme known to minimize the sum of per key estimation variances when sampling pre-aggregated data, we obtain SA[VarOpt] and its stream variant SA[VarOpt](stream). Although we show that there is no streaming algorithm that achieves the same minimum on unaggregated streams, SA[VarOpt] has the novel desirable property that the variance is progressively closer to the pre-aggregated minimum when applied to a "more" aggregated data set. We provide an efficient implementation of our algorithm. Extensive evaluation on synthetic and real-life data sets shows that SA[VarOpt](stream) significantly outperforms all other streaming based estimates for unaggregated data in terms of accuracy for a given summary size. Even larger performance improvements are established for the generic SA[VarOpt] applicable to distributed data sets.

11. REFERENCES

[1] M. T. Chao. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982.

[2] S. Chaudhuri, R. Motwani, and V.R. Narasayya. On random sampling over joins. In *Proc. ACM SIGMOD Conference*, pages 263–274, 1999.

[3] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Algorithms and estimators for accurate summarization of Internet traffic. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC)*, 2007.

[4] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Algorithms and estimators for accurate summarization of Internet traffic. Manuscript, 2007.

[5] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Sketching unaggregated data streams for subpopulation-size queries. In *Proc. of the 2007 ACM Symp. on Principles of Database Systems (PODS 2007)*. ACM, 2007.

[6] E. Cohen, N. Duffield, H. Kaplan, C. Lund, and M. Thorup. Stream sampling for variance-optimal estimation of subset sums. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 2009.

[7] E. Cohen, N. Duffield, C. Lund, and M. Thorup. Confident estimation for multistage measurement sampling and aggregation. In *ACM SIGMETRICS*, 2008. June 2-6, 2008, Annapolis, Maryland, USA.

[8] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *Proc. 26th ACM PODC*, 2007.

[9] E. Cohen and H. Kaplan. Tighter estimation using bottom-k sketches. In *Proceedings of the 34th VLDB Conference*, 2008.

[10] N. Duffield and C. Lund. Predicting resource usage and estimation accuracy in an ip flow measurement collection infrastructure. In *ACM SIGCOMM Internet Measurement Workshop*, 2003. Miami Beach, Fl, October 27-29, 2003.

[11] N. Duffield, M. Thorup, and C. Lund. Priority sampling for estimating arbitrary subset sums. *J. Assoc. Comput. Mach.*, 54(6), 2007.

[12] N.G. Duffield, C. Lund, and M. Thorup. Learn more, sample less: control of volume and variance in network measurements. *IEEE Transactions on Information Theory*, 51(5):1756–1775, 2005.

[13] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of the ACM SIGCOMM'02 Conference*. ACM, 2002.

[14] C.T. Fan, M.E. Muller, and I. Rezucha. Development of sampling plans by using sequential (item by item) selection techniques and digital computers. *J. Amer. Stat. Assoc.*, 57:387–402, 1962.

[15] P. Gibbons and Y. Matias. New sampling-based summary statistics for improving approximate query answers. In *SIGMOD*. ACM, 1998.

[16] J. Hájek. Asymptotic theory of rejective sampling with varying probabilities from a finite population. *The Annals of Mathematical Statistics*, 35(4):1491–1523, 1964.

[17] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952.

[18] T. Johnson, S. Muthukrishnan, and I. Rozenbaum. Sampling algorithms in a stream operator. In *Proc. ACM SIGMOD*, pages 1–12, 2005.

[19] D.E. Knuth. *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*. Addison-Wesley, 1969.

[20] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security and Privacy Magazine*, 1(4):33–39, 2003.

[21] The Netflix Prize. `http://www.netflixprize.com/`.

[22] Cisco NetFlow. `http://www.cisco.com/warp/public/732/Tech/netflow`.

[23] E. Ohlsson. Sequential poisson sampling. *J. Official Statistics*, 14(2):149–162, 1998.

[24] B. Rosén. Asymptotic theory for successive sampling with varying probabilities without replacement, i. *The Annals of Mathematical Statistics*, 43(2):373–397, 1972.

[25] B. Rosén. Asymptotic theory for order sampling. *J. Statistical Planning and Inference*, 62(2):135–158, 1997.

[26] S. Sampath. *Sampling Theory and Methods*. CRC press, 2000.

[27] R. Singh and N. S. Mangat. *Elements of survey sampling*. Springer-Verlag, New York, 1996.

[28] A. B. Sunter. List sequential sampling with equal or unequal probabilites without replacement. *Applied Statistics*, 26:261–268, 1977.

[29] M. Szegedy and M. Thorup. On the variance of subset sum estimation. In *Proc. 15th ESA, LNCS 4698*, pages 75–86, 2007.

[30] Y. Tillé. An elimination procedure for unequal probability sampling without replacement. *Biometrika*, 83(1):238–241, 1996.

[31] Y. Tillé. *Sampling Algorithms*. Springer-Verlag, New York, 2006.

[32] J.S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

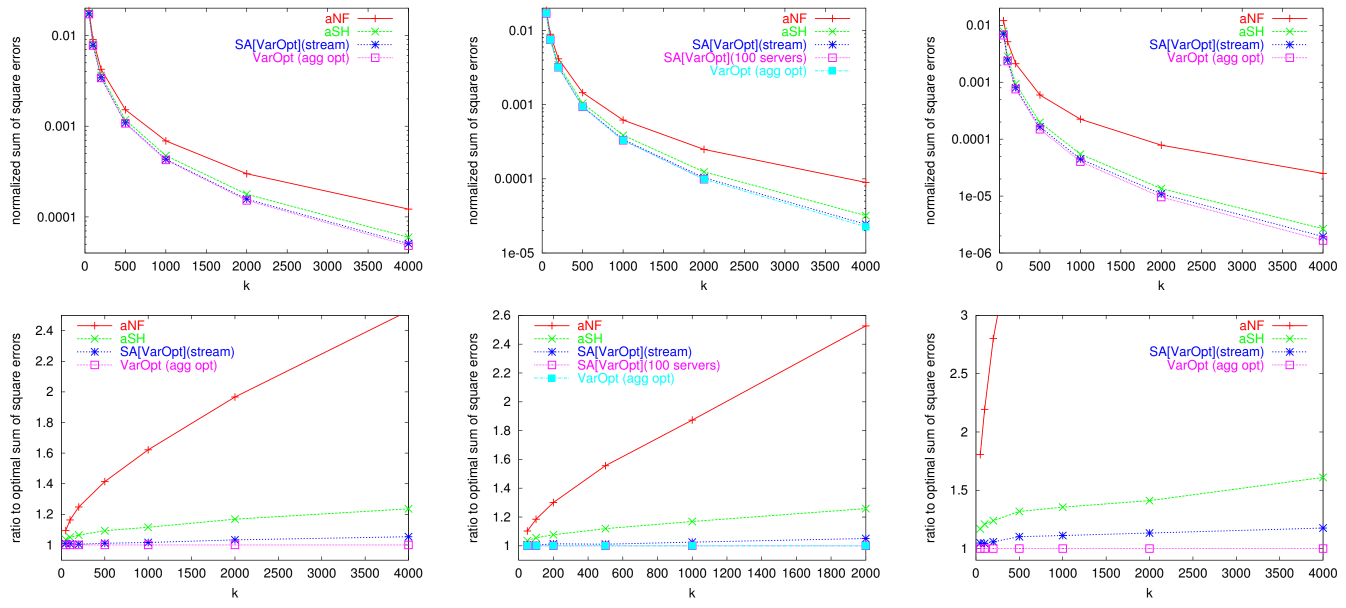

Figure 9: IP packet header streams with keys according to IP address pairs on the peering data (left), destination IP addresses on the peering data (middle), and IP address pairs on the campus data (right). Sweeping the sample size k. Top: Normalized average sum of square errors. Bottom: The ratio of the average sum of square errors to $\Sigma V[\text{VarOpt}_k]$.

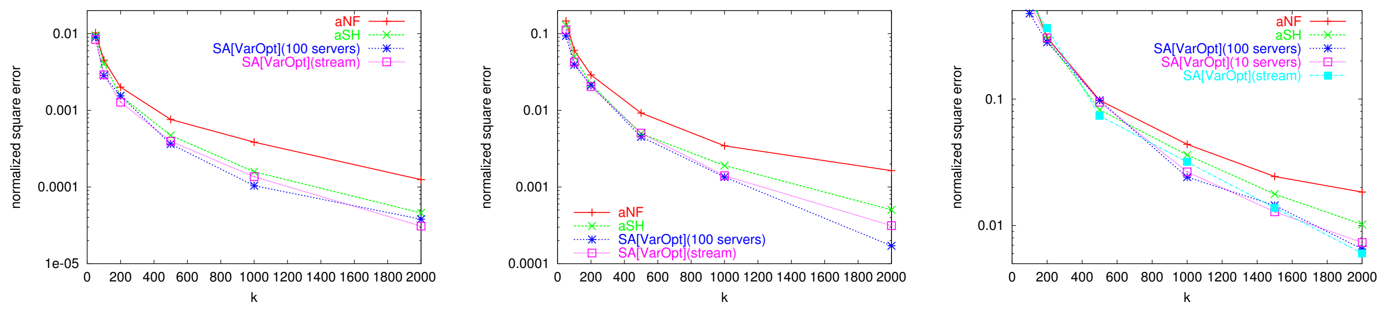

Figure 10: Left and middle: Normalized square error, averaged over 100 runs, on the total bytes of subpopulations of destination IP addresses from the peering IP packet header stream. Estimates gleaned from SA[VarOpt](stream), SA[VarOpt$_k$](servers) (on 100 servers with random assignment of flows to servers), aSH, and aNF summaries. Subpopulations selection: destinations that support web flows (left) and p2p flows (middle). Right: Normalized square error, averaged over 100 runs, on the total requests for movies released in 1991 (Netflix dataset) (316 movies, 1.7×10^6 requests). Estimates gleaned from SA[VarOpt](stream), aSH, aNF, and SA[VarOpt$_k$](servers) (10 and 100 servers) summaries.

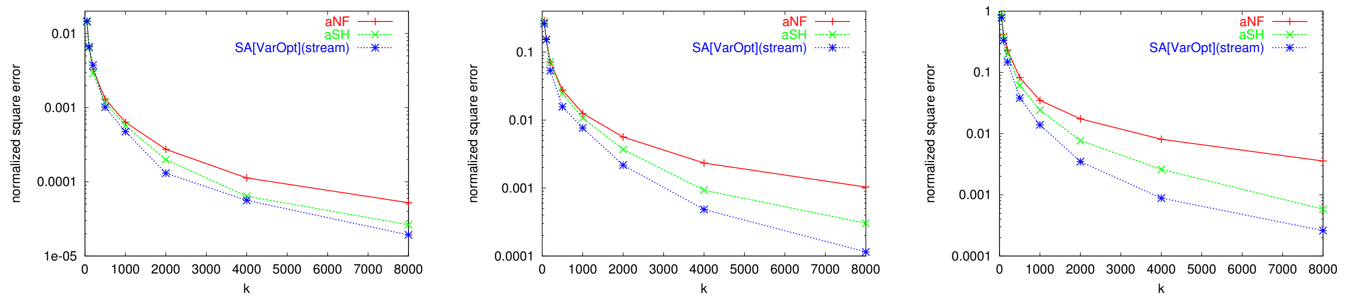

Figure 11: Normalized square error, averaged over 100 runs, on the total bytes of subpopulations of flows from the peering IP packet header stream. Estimates gleaned from SA[VarOpt](stream), aSH, and aNF summaries. Subpopulations: web flows (left), p2p flows (middle), multimedia flows (right).