On the algebra of data sketches

Jakub Lemiesz
Wrocław University of Science and Technology
Wrocław, Poland
jakub.lemiesz@pwr.edu.pl

ABSTRACT

We consider the problem of designing a distributed data sketch for scenario in which data stream is observed by many independent network nodes. We require that a sketch apart from being computationally and memory efficient should also be mergeable in a way that mimics set theory operations on related data sets.

For example, when monitoring network traffic, one may consider how many distinct packets passed through a given node (sum of sets for different time windows) or passed through two given nodes (intersection of sets from two locations) and what is their total size (intersection of weighted sets).

In this paper we propose a sketch that allows to efficiently summarize sets constructed from a sequence of set theory operations. We also provide an analytical control over the trade-off between the accuracy and storage/computational requirements. In comparison to the previous works the proposed solution 1) allows the weights of elements, 2) allows performing set theory operations simultaneous on a large number of sketches, 3) does not require computationally expensive numerical calculations and guarantees low overheads.

PVLDB Reference Format:
Jakub Lemiesz. On the algebra of data sketches. PVLDB, 14(9): 1655 - 1667, 2021.
doi:10.14778/3461535.3461553

1 INTRODUCTION

1.1 Cardinality estimation

For massive data streams, often containing millions of elements, solutions based on storing all data and analyzing them afterwards are usually far too costly in terms of time as well as memory consumption. The classic example is the problem of counting the number n of distinct stream elements. It is known that, without some additional knowledge about the nature of the data, storage linear in n is required for the precise answer [2].

Approximate counting algorithms with sublinear storage requirements have been intensively developed and analyzed since 1985 when Flajolet and Martin published the landmark article [25]. Flajolet's algorithm and subsequent solutions employ hash functions, which provide an easily reproducible substitute of randomness and allow to assign identical elements to the same pseudorandom value. More precisely, one may treat an input stream as a multiset $\mathfrak{M} = (\mathbb{U}, f)$, where $\mathbb{U}$ denotes underlying set with unknown cardinality $|\mathbb{U}| = n$ and function $f : \mathbb{U} \to \mathbb{N}_{\geq 1}$ determines the multiplicity of each element in $\mathbb{U}$. Then, by applying a hash function $h : \mathbb{U} \to \{0, 1\}^b$ one transforms multiset $\mathfrak{M}$ into what can be regarded as a set of outcomes of n independent uniform random variables (cf. [29]). In the last step, based on some characteristic patterns in those outcomes, different estimators of unknown parameter n can be proposed. The idea is that hashing to a small number of bits b suffices to estimate vary large cardinalities n.

In the literature there are generally two views on the set of outcomes $h(\mathfrak{M})$ – a continuous and discrete view. In the continuous view the outcomes are considered as random real numbers and estimations are based on their order statistics. The best known algorithm in this category is KMV (k-minimum values, see [4, 10]), which is based on the kth order statistic of n uniformly distributed random variables. Other algorithms that belong to this group are discussed for example in [26]. They all attain a standard error of approximately $1/\sqrt{m}$ when m elements are stored in the sketch. The algorithm that has been employed in this paper also belongs to this category.

In the discrete view the outcomes are considered as strings of bits and cardinality estimation is based on the bit-pattern observation, e.g. one may keep tracking the position of the leftmost 1-bit. In this group one has Flajolet's pioneering algorithm [25] or more recent LogLog [22], HyperLogLog [24] and HyperLogLog++ [27]. The standard error of these algorithms is also close to $1/\sqrt{m}$ for m stored elements. However, memory required to store a single element is logarithmically smaller than for algorithms based on order statistics (cf. [24]). Therefore, algorithms in this category are much more space-efficient. For instance, HyperLogLog can estimate cardinalities up to 10^9 with a typical accuracy of 2% and only 1.5 kilobytes of memory.

1.2 Weighted cardinality

Cardinality estimation problem can be generalized to a weighted version, where each element i is associated with a fixed weight $w_i \in \mathbb{R}_+$. In the generalized problem the goal is to estimate the total sum of weights for all unique elements present in a stream. The authors of [18] have shown that every cardinality estimation algorithm based on the extreme order statistics (i.e. based on storing the minimum or maximum values of random variables) can be transformed into the weighted counterpart.

In this paper we present a solution founded on storing the minimum of exponentially distributed random variables. The associated weighted cardinality estimator has been derived by maximizing the likelihood function in [3]. At the beginning of Section 3 we briefly show how this estimator, presented further in the formula (4), can be derived in a simpler way and how it can be used to construct more complex estimators.

Proceedings of the VLDB Endowment, Vol. 14, No. 9 ISSN 2150-8097.
doi:10.14778/3461535.3461553

1.3 Operation on data sketches

Cardinality estimation algorithms and – more generally – frequency estimation algorithms (see e.g. [19]) are founded on data sketches that concisely summarize corresponding data sets. The possibility of simulating the set theory operations based on such data sketches has been recently considered in a few papers (e.g. [1, 5, 20, 23, 34]).

As described in [20] simulating set operations with KMV sketches is quite efficient. Given two KMV sketches $S_{\mathbb{A}}$ and $S_{\mathbb{B}}$ that represent sets $\mathbb{A}$ and $\mathbb{B}$ by storing k smallest of the observed hashes, one can create sketch $S_{\mathbb{A}\cup\mathbb{B}}$ representing the union $\mathbb{A} \cup \mathbb{B}$ only by storing the k smallest values from set $S_{\mathbb{A}} \cup S_{\mathbb{B}}$. Based on sketch $S_{\mathbb{A}\cup\mathbb{B}}$ we get an estimate of $|\mathbb{A} \cup \mathbb{B}|$, which multiplied by Jaccard similarity $J(\mathbb{A}, \mathbb{B})$ gives an estimate of $|\mathbb{A} \cap \mathbb{B}|$. Jaccard similarity $J(\mathbb{A}, \mathbb{B})$ of sets $\mathbb{A}$ and $\mathbb{B}$ can be naturally estimated as the cardinality of set $S_{\mathbb{A}\cup\mathbb{B}} \cap S_{\mathbb{A}} \cap S_{\mathbb{B}}$ divided by k. The operations can be generalized to a larger number of sketches and sketches can be augmented with multiplicity counters to keep the count of all occurrences of an element in a multiset (see AKMV in [5]).

The authors of [34] consider Streaming KMV sketches that augment basic KMV sketches with a running estimate of the cardinality to reduce the variance of the estimator by half. Two estimation procedures for union and intersection operations based on new sketches are proposed. The first involves the computationally costly maximum likelihood optimization, the second – more efficient – averages a number of unbiased or consistent estimators.

An important property of the HyperLogLog algorithm is that given two sketches $S_{\mathbb{A}}$ and $S_{\mathbb{B}}$ one can create sketch $S_{\mathbb{A}\cup\mathbb{B}}$ as a pointwise maximum $S_{\mathbb{A}\cup\mathbb{B}}[i] = \max(S_{\mathbb{A}}[i], S_{\mathbb{B}}[i])$ and the resulting sketch is identical to the sketch constructed directly from set $\mathbb{A} \cup \mathbb{B}$. Unfortunately, handling intersections is much more troublesome. A standard approach is to estimate the size of intersection based on sketches $S_{\mathbb{A}}$, $S_{\mathbb{B}}$, $S_{\mathbb{A}\cup\mathbb{B}}$ and principle $|\mathbb{A} \cap \mathbb{B}| = |\mathbb{A}| + |\mathbb{B}| - |\mathbb{A} \cup \mathbb{B}|$. However, as the number of set operations increases, this technique based on the inclusion-exclusion principle becomes very cumbersome, expensive and inaccurate (see [5, 20]).

A more sophisticated method of using HyperLogLog sketches to estimate the cardinality of the union, intersection and relative complement has been proposed in [23]. The method is based on the maximum likelihood estimation and although gives a more precise estimate, it requires a substantial computational effort.

In [1] the authors show that the algorithms connected to the frequency estimation problem can also produce mergeable sketches.

Let us emphasize that none of the above solutions addresses the problem with regard to the weighted cardinality as they are designed for discrete domains (i.e. cardinality or frequency estimation). Moreover, since the main focus was on the statistical properties of proposed estimators, these solutions often require computationally expensive numerical calculations and in many scenarios are impractical. In particular, estimators proposed in [23] and [34] have no explicit formulas as they are based on the numerical optimization. In consequence – considering their discrete nature or mentioned performance limitations – existing solutions seem too limited for a variety of practical application domains (see Section 5 and 6). Finally, problems such as creating reusable sketches for intersection and relative complement, operating simultaneously on many sketches or secure transfer of sketches have not been well studied.

1.4 Problem formulation

In this paper we propose a sketching framework that for unweighted sets matches the results achieved by KMV sketches and addresses the shortcomings of existing solutions mentioned above by meeting the following requirements.

(1) Allow weighted sets while providing low storage and computational overhead for both creating the sketch and using them in the estimation process. Allowing weights is justified by various applications (see Section 5) and enables easy construction of numerous estimators (see Section 3).

(2) Allow for efficient simulation of set theory operations simultaneously on a large number of sketches. The result of these operations can be stored in reusable sketches (see Section 4).

(3) Allow for efficient and secure transfer of information in a distributed environment. Proposed sketches store minima, which enables using information spreading techniques with low communication overhead (see *extrema propagation* [17]). Moreover, the cryptographic mechanisms can be naturally incorporated into the framework (see Section 5).

2 DATA SKETCH

Let $\mathfrak{M} = (\mathbb{U}, f)$ be a multiset in which function $f : \mathbb{U} \to \mathbb{N}_{\geq 1}$ determines the multiplicity of each element in $\mathbb{U}$. We assume that each element $e \in \mathbb{U}$ can be described as a tuple of $d + 1$ elements $e = (i, \lambda_{i,1}, \ldots, \lambda_{i,d})$, where i is a unique identifier of the element e and $\lambda_{i,j} \in \mathbb{R}_+$ describes the jth feature of e. For the sake of simplicity we assume that if there are n distinct elements in $\mathfrak{M}$, each of them has a unique identifier $i \in \{1, 2, \ldots, n\}$. In practice identifier i may be any unique sequence of an appropriate length.

In this paper we focus on Algorithm 1 that creates a sketch of a data stream $\mathfrak{M}$. For each element in the stream $\mathfrak{M}$ we calculate $d \times m$ hash values. The parameter d denotes the number of features of each element and m is the number of independent experiments that controls the precision of estimates one can get from the sketch.

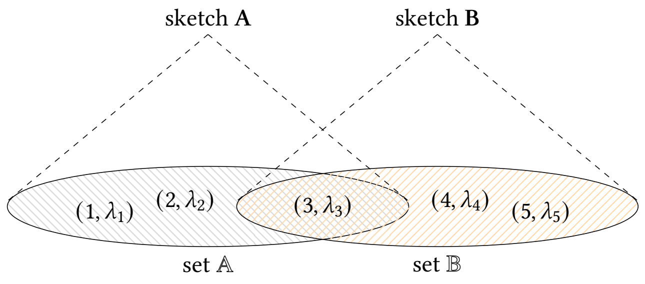


Figure 1: Sets of elements with weights represented by sketches. Based on sketches A and B we would like to estimate the total weights of sets $\mathbb{A}$ and $\mathbb{B}$. Moreover, based on sketches A and B we would like to generate sketches for sets $\mathbb{A} \cup \mathbb{B}, \mathbb{A} \cap \mathbb{B}, \mathbb{A} \setminus \mathbb{B}$. In many scenarios each element can be observed multiple times or the information can come to the observer through different paths (e.g. as a packet or radio signal). For this reason, we model the input as multisets $(\mathbb{A}, f)$ and $(\mathbb{B}, g)$.

Algorithm 1 CreateSketch($\mathfrak{M}$, d, m)

```
Initialization:
1:  set each of $d \times m$ positions of sketch $\mathbf{M}$ to $\infty$

Upon arrival of an element $(i, \lambda_{i,1}, \ldots, \lambda_{i,d}) \in \mathfrak{M}$ :
2:  for all $j \in \{1, 2, \ldots, d\}$ do
3:      for all $k \in \{1, 2, \ldots, m\}$ do
4:          $u \leftarrow h(i^\frown j^\frown k)$
5:          $\mathbf{M}[j,k] \leftarrow \min\left\{\mathbf{M}[j,k], -\frac{\ln u}{\lambda_{i,j}}\right\}$
6:      end for
7:  end for

Upon request, at any time:
8:  Return: M
```

For the purposes of a formal analysis we assume that the hash function $h : \{0,1\}^u \to 0.\{0,1\}^v$, $u, v \in \mathbb{N}_+$ is truly random and its output values are uniformly distributed in the unit interval. By $i^\frown j$ we denote concatenation of two numbers, each represented by the sequence with a fixed length (e.g. by 32-bit binary representation).

Algorithm 1 is a very simple procedure of aggregating the data from a stream $\mathfrak{M}$ – its simplicity is the precondition for stream processing applications. As always in this case, the main problem is how efficient is the sketch for storing the information that we need to keep in the context of a given problem.

In Section 3 we analyze estimators that can be used to infer from a sketch $\mathbf{M}$ information about the distribution of each single feature j as well as about the relations between these features.

In Section 4 we show how different sketches can be merged to mimic set theory operations on the corresponding data streams.

3 DERIVATION OF ESTIMATORS

3.1 Single experiment

We start with the case that each element of multiset $\mathfrak{M}$ has a single feature, that is, it is a pair (i, λ_i). Our goal is to estimate the value of $\Lambda = \sum_{i=1}^{n} \lambda_i$. Suppose also that for each element we carry out a single experiment (generate a single hash value). Thus we have $d = 1$ and $m = 1$ and the sketch $\mathbf{M}$ is just a single value.

Based on our assumptions on the hash function, values $h(i^\frown 1^\frown 1)$ generated for $i = 1, \ldots, n$ in line 4 of Algorithm 1 can be considered as uniformly distributed independent random variables. Therefore random variables $S_i = -\ln h(i^\frown 1^\frown 1) \,/\, \lambda_i$ obtained in line 5 of Algorithm 1 are also mutually independent. From the inverse transform sampling theorem ([21], p. 29) follows that S_i is exponentially distributed: $S_i \sim Exp(\lambda_i)$. The value stored in $\mathbf{M}$ after the whole stream has been processed is $\mathbf{M} = \min\{S_1, \ldots, S_n\}$ (see line 5 of Algorithm 1). Since $S_1, \ldots, S_n$ are independent we have

$$\Pr(\mathbf{M} \geq x) = \prod_{i=1}^{n} \Pr(S_i \geq x) = e^{-(\lambda_1 + \ldots + \lambda_n)x} = e^{-\Lambda x} . \qquad (1)$$

Thus $\mathbf{M}$ also follows the exponential distribution: $\mathbf{M} \sim Exp(\Lambda)$. Since $\mathbb{E}(\mathbf{M}) = 1/\Lambda$ we could try to estimate Λ by $\overline{\Lambda} := 1/\mathbf{M}$. The problem with this approach is that $\mathbb{E}(1/\mathbf{M})$ is unbounded. Indeed, $\mathbb{E}(1/M) = \int_0^\infty \frac{1}{x}\Lambda e^{-\Lambda x} dx$. This is one of the reasons why the sketch should consist of more than one experiment.

3.2 Multiple independent experiments

Suppose $d = 1$ and for each element we carry out $m \geq 2$ independent experiments. Thus, a sketch $\mathbf{M}$ is now a single row $\mathbf{M} = (\mathbf{M}_1, \ldots, \mathbf{M}_m)$. Note that $\mathbf{M}_k \sim Exp(\Lambda)$ and $\mathbf{M}_1, \ldots, \mathbf{M}_m$ are independent since a different value of parameter k is used for generating hash values in each experiment (line 4 of Algorithm 1).

The sum of m independent exponentially distributed random variables with the same mean Λ follows the gamma distribution:

$$G_m := \sum_{k=1}^{m} \mathbf{M}_k \sim \Gamma(m, \Lambda) ,$$

where $m \in \{1, 2, 3, \ldots\}$ and $\Lambda \in (0, \infty)$ represent the shape and the rate parameters. Thus, for random variable G_m we have

$$\mathbb{E}(G_m) = \frac{m}{\Lambda} , \quad \mathbb{V}\mathrm{ar}(G_m) = \frac{m}{\Lambda^2} . \qquad (2)$$

The inverse of random variable G_m has the inverse gamma distribution $1/G_m \sim \Gamma^{-1}(m, \Lambda)$ (see e.g. [37]) and we have

$$\mathbb{E}(1/G_m) = \tfrac{\Lambda}{m-1}, \quad \mathbb{V}\mathrm{ar}(1/G_m) = \tfrac{\Lambda^2}{(m-1)^2(m-2)} \qquad (3)$$

where the first equation holds for $m \geq 2$ and second for $m \geq 3$.

From the first equation in (3) we can conclude that

$$\overline{\Lambda} := \frac{m-1}{G_m} \qquad (4)$$

is an unbiased estimator of Λ for $m \geq 2$, namely $\mathbb{E}(\overline{\Lambda}) = \Lambda$. Moreover, by the second equation in (3) for $m \geq 3$ we get

$$\mathbb{V}\mathrm{ar}\left(\overline{\Lambda}/\Lambda\right) = \frac{1}{m-2} . \qquad (5)$$

To derive a convenient two-sided concentration bound for $\overline{\Lambda}$ the Cramér's Theorem can been used (see [32], Lemma 2).

Let us remark that the estimator $\overline{\Lambda}$ is the harmonic mean of estimations $1/\mathbf{M}_1, \ldots, 1/\mathbf{M}_m$ (up to the factor $m-1$ instead of m). Therefore, even if a part of these estimations heavily overestimate Λ, it will not significantly affect the value of $\overline{\Lambda}$ as it is dominated by the smallest of them.

Let us also remark that if a feature of the elements observed from the data stream is always 1, namely $\lambda_i = 1$ for all $i \in \{1, \ldots, n\}$ and thus $\Lambda = n$, it can be shown that $\overline{\Lambda}$ is a maximum likelihood estimator of parameter n, where n is the number of distinct elements in a multiset $\mathfrak{M}$ (see [3]).

3.3 Independent features

Now we focus our attention on the most essential case that elements have $d \geq 1$ features and $m \geq 3$. Note that the jth row of sketch $\mathbf{M}$ corresponds to m experiments for feature j (line 5 of Algorithm 1). Experiments in different rows can be regarded as independent since j is one of the arguments of the hash function (line 4 of Algorithm 1). Then, by analogy to the estimator (4), for the sketch $\mathbf{M}$ we can obtain a vector of d independent estimators $\overline{\mathbf{\Lambda}} = (\overline{\Lambda}_1, \ldots, \overline{\Lambda}_d)$ where $\overline{\Lambda}_j := \frac{m-1}{\sum_{k=1}^{m} \mathbf{M}_{j,k}}$ is an unbiased estimator of $\Lambda_j = \sum_{i=1}^{n} \lambda_{i,j}$.

Independence of the estimators $\overline{\Lambda}_j$ turn out to be very useful for constructing other estimators. In particular, it can be used to estimate the average value, the variance and other moments of a feature. These in turn can be used for more in-depth inquiries about the distribution (e.g. based on tail and concentration bounds).

For instance, assume that we observe elements (i, λ_i) and our goal is to infer about the distribution from which the values λ_i are sampled. Our trick is to encode each element (i, λ_i) as a tuple $(i, \lambda_{i,0}, \lambda_{i,1}, \lambda_{i,2})$, where for all elements $\lambda_{i,0} = 1$, $\lambda_{i,1} = \lambda_i$ and $\lambda_{i,2} = (\lambda_i)^2$. Then, we use $\overline{\Lambda}_0, \overline{\Lambda}_1, \overline{\Lambda}_2$ to estimate n, $\Lambda = \sum_{i=1}^{n} \lambda_i$ and $\sum_{i=1}^{n} (\lambda_i)^2$, respectively, using the method described above. Then, to estimate the average $A = \Lambda/n$ we compute the ratio $\overline{\Lambda}_1 / \overline{\Lambda}_0$. Since $1 / \overline{\Lambda}_0$ and $\overline{\Lambda}_1$ are independent random variables with gamma and inverse gamma distribution, by using the expected values presented in formulas (2) and (3) we get $\mathbb{E}(\overline{\Lambda}_1 / \overline{\Lambda}_0) = \frac{\Lambda}{m-1} \frac{m}{n}$. Therefore we can define unbiased estimator of A as

$$\bar{A} := \frac{m-1}{m} \cdot \frac{\overline{\Lambda}_1}{\overline{\Lambda}_0} \,. \tag{6}$$

Knowing the variances of random variables $1/\overline{\Lambda}_0$ and $\overline{\Lambda}_1$ presented in formulas (2) and (3) and based on their independence we can also easily show that $\mathbb{V}\mathrm{ar}(\bar{A}/A) = 2/m + O\left(1/m^2\right)$.

Similarly we can construct and analyze an estimator for any raw or central moment. However, in case of central moments one need to be careful while using standard formulas. For example, when the variance $\mathbb{V}\mathrm{ar}\,(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$ is small and estimates for $\mathbb{E}(X^2)$ and $\mathbb{E}(X)^2$ are large, then the variance estimation obtained from their subtraction is unreliable. Fortunately, one of several stable single-pass procedures for computing central moments can be applied based on the sketch $\mathbf{M}$ (e.g. the Welford's algorithm [9]).

In general, independence of experiments for different features enables analyzing their relationship. For example, in exactly the same way as for the average Λ/n one can derive unbiased estimator of the ratio $\Lambda_{j_1}/\Lambda_{j_2}$ for any two features j_1 and j_2. Moreover this independence allows for merging the features. Namely, we can take the position-wise minimum of two rows of the sketch $\mathbf{M}$ corresponding to features j_1 and j_2 to obtain a single row corresponding to the scenario in which elements of the form $(i^\frown j_1, \lambda_{i,j_1})$ and $(i^\frown j_2, \lambda_{i,j_2})$, $j_1 \neq j_2$ are observed (i.e. two different features of the same element are treated as as the same feature of different elements). For the row obtained one can use estimator (4) and thereby estimate $\Lambda_{j_1} + \Lambda_{j_2}$.

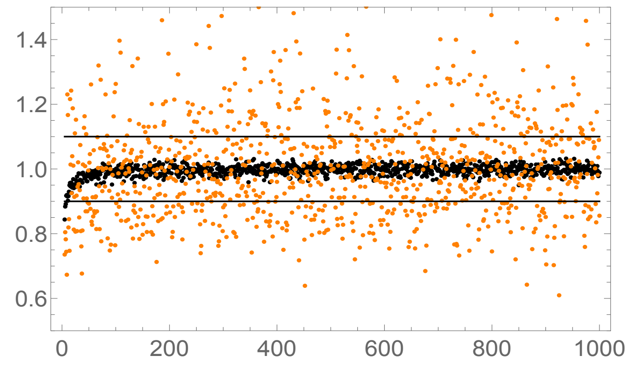
3.4 Dependent features

Coupling the experiments for different features may also have benefits. Let us recall that for an element $\left(i, \lambda_{i,1}, \ldots, \lambda_{i,d}\right)$ the values

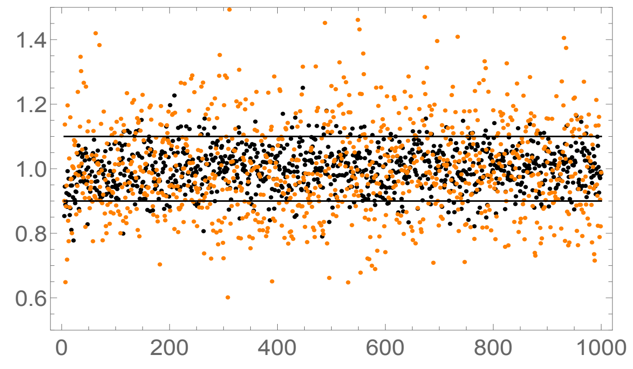
$$-\ln h\,(i^\frown j^\frown k)\ /\ \lambda_{i,j} \tag{7}$$

generated in line 5 of Algorithm 1 are computed in a deterministic way. Therefore only the first occurrence of $\left(i, \lambda_{i,1}, \ldots, \lambda_{i,d}\right)$ can change the sketch. However, for two elements with the same values of features but starting with different ID's i_1, i_2 the values in (7) generated for these elements are independent as i_1 and i_2 are taken as part of the input for the hash function. Similarly, since the feature number j is a part of the input to the hash function, values generated for different features are independent. Similarly, taking the experiment number k guarantees the independence of experiments for a given feature.

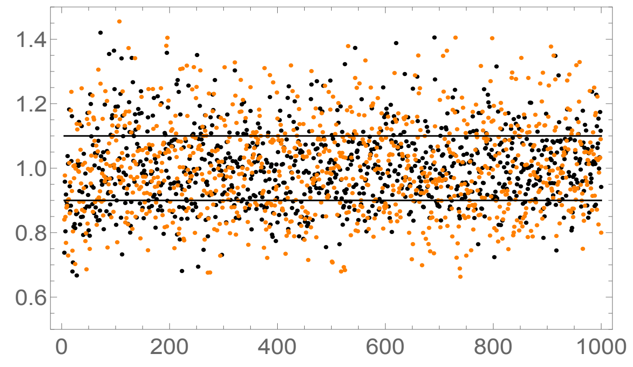
Somewhat surprisingly, if we remove the feature number j from the input to the hash function, simulations show that some estimations become significantly more precise.



(a) $\lambda_1, \ldots, \lambda_n \sim \mathcal{N}(1, 0.1)$.



(b) $\lambda_1, \ldots, \lambda_n \sim \mathcal{U}(0, 1)$.



(c) $\lambda_1 = \ldots = \lambda_{n-1} = 1$ **and** $\lambda_n = 1000$.

Figure 2: Figures 2a - 2c present simulation results comparing normalize estimations of the average $\bar{A}/A$ (orange dots) based on independent experiments and $\hat{A}/A$ (black dots) based on dependent experiments for $m = 100$. For each number of elements $n \in \{2, 3, \ldots, 1000\}$ we make a single experiment. Results for different values of $\lambda_1, \lambda_2, \ldots, \lambda_n$ presented on each figure show that $\hat{A}$ in most cases is much more concentrated. By $\mathcal{N}$ and $\mathcal{U}$ we denote the normal and uniform distribution. The black horizontal lines symbolize the 10% error boundaries. The difference between $\bar{A}$ and $\hat{A}$ is particularly visible when averaged values are similar. Note however that the presence of significant outliers can close the difference between the compared estimators (see Figure 2c).

For example, let us consider the average estimator $\bar{A}$ based on independent estimators of n and Λ defined by formula (6). By removing the feature number j from the input to the hash function we can generate correlated values

$$-\ln h(i^\frown k)\ /\ 1 \quad \text{and} \quad -\ln h(i^\frown k)\ /\ \lambda_i$$

respectively for the cardinality estimator $\overline{\Lambda}_0$ and sum estimator $\overline{\Lambda}_1$. By this change we obtain a similar estimator $\hat{A}$ that has much higher precision compared to the original estimator $\bar{A}$. The experimental comparison of two estimators $\bar{A}$ and $\hat{A}$ is presented in the Figure 2. The significant difference in the concentration of $\bar{A}$ and $\hat{A}$ is related to the fact that the hashes generated for $\overline{\Lambda}_0$ and $\overline{\Lambda}_1$ are correlated – if they are extremely close to zero, then they simultaneously influence *both* estimators. While the intuition behind this phenomenon is clear the formal analysis of the properties of the estimator $\hat{A}$ seems to be much more complex.

4 OPERATIONS ON DATA SKETCHES

Consider two data sketches **A** and **B** generated by Algorithm 1 for multisets $(\mathbb{A}, \cdot)$ and $(\mathbb{B}, \cdot)$. Note that the number of occurrences of a given element in a multiset does not affect the values in the sketch. This is due to fact that in Algorithm 1 we use a deterministic hash function and only the first occurrence of an element matters. In this section we show how **A** and **B** can be used to estimate the results of set theory operations on $\mathbb{A}$ and $\mathbb{B}$. For brevity of exposition let us assume that the elements of $\mathbb{A}$ and $\mathbb{B}$ have only one feature, namely they are of the form (i, λ_i). In this case sketches are vectors of length m: $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m)$, $\mathbf{B} = (\mathbf{B}_1, \mathbf{B}_2, \ldots, \mathbf{B}_m)$. Recall that values stored in $\mathbf{A}_k$ and $\mathbf{B}_k$ correspond to exponentially distributed random variables $\mathbf{A}_k \sim Exp(|\mathbb{A}|_w)$ and $\mathbf{B}_k \sim Exp(|\mathbb{B}|_w)$, where $|\mathbb{A}|_w := \sum_{(i,\lambda_i)\in\mathbb{A}} \lambda_i$.

4.1 Union of sketches

Our goal now is to estimate the value of $|\mathbb{A} \cup \mathbb{B}|_w$ based on **A** and **B**. The solution is simple and efficient: we create sketch $\mathbf{A} \uplus \mathbf{B}$ using the point-wise minimum: $\mathbf{A} \uplus \mathbf{B} := (\min(\mathbf{A}_1, \mathbf{B}_1), \ \ldots, \ \min(\mathbf{A}_m, \mathbf{B}_m))$. Since $\min(\min(\mathbb{X}), \min(\mathbb{Y})) = \min(\mathbb{X} \cup \mathbb{Y})$ for two sets of numbers $\mathbb{X}$ and $\mathbb{Y}$, then $\mathbf{A} \uplus \mathbf{B}$ corresponds to the sketch we would get by observing elements of $\mathbb{A} \cup \mathbb{B}$. Thus, $\min(\mathbf{A}_k, \mathbf{B}_k) \sim Exp(|\mathbb{A} \cup \mathbb{B}|_w)$ and as follows from definition (4) for $m \geq 2$ we can define unbiased estimator of $|\mathbb{A} \cup \mathbb{B}|_w$ as

$$\overline{U}(\mathbf{A}, \mathbf{B}) := \frac{m-1}{\sum_{k=1}^{m} \min(\mathbf{A}_k, \mathbf{B}_k)} \ . \tag{8}$$

Clearly, one can generalize this method for any number of sketches $\overline{U}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots)$ by replacing summands in (8) by $\min(\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k, \ldots)$. This ease of processing sums of sets might be a killer application of data sketches – distributed summations of data elements with no double counting (see Section 5).

Note also that considered family of sketches is closed under proposed union operation, so sketch $\mathbf{A} \uplus \mathbf{B}$ representing set $\mathbb{A} \cup \mathbb{B}$ is a full-fledged sketch and can be stored and used in other operations.

4.2 Jaccard Similarity

Jaccard similarity of sets is defined as $J(\mathbb{X}, \mathbb{Y}) := |\mathbb{X} \cap \mathbb{Y}| / |\mathbb{X} \cup \mathbb{Y}|$. The generalizations of Jaccard similarity that take into account weights of elements have been proposed several times over many years (for a long list of papers see [13]). We define weighted Jaccard similarity on sets $\mathbb{A}$ and $\mathbb{B}$ as

$$J_w(\mathbb{A}, \mathbb{B}) := \frac{|\mathbb{A} \cap \mathbb{B}|_w}{|\mathbb{A} \cup \mathbb{B}|_w}. \tag{9}$$

Jaccard similarity can be efficiently estimated by the well-known MinHash algorithm (see e.g. [6]). Some interesting techniques for estimating the weighted Jaccard that introduce weights into the computation of MinHash have been proposed e.g. in [14, 28, 33].

We will show that in a similar way to how MinHash works sketches **A** and **B** of respectively the set $\mathbb{A}$ and $\mathbb{B}$ can be used to easily estimate the value of $J_w(\mathbb{A}, \mathbb{B})$.

Lemma 4.1. *Let* $\mathbf{A}_k$ *and* $\mathbf{B}_k$ *be the kth elements of the sketches* **A** *and* **B** *created for sets* $\mathbb{A}$ *and* $\mathbb{B}$. *Then* $J_w(\mathbb{A}, \mathbb{B}) = \Pr(\mathbf{A}_k = \mathbf{B}_k)$.

Proof. Let

$$\mathbb{S}_A = \left\{ S_{i,k} \ : \ (i, \lambda_i) \in \mathbb{A} \ \wedge \ S_{i,k} = \frac{-\ln h(i \frown k)}{\lambda_i} \right\} .$$

Note that we have $\mathbf{A}_k = \min(\mathbb{S}_A)$. Analogously we define $\mathbb{S}_B$ and get $\mathbf{B}_k = \min(\mathbb{S}_B)$. Note also that if $(i, \lambda_i) \in \mathbb{A} \cap \mathbb{B}$, then $S_{i,k} \in \mathbb{S}_A \cap \mathbb{S}_B$. Apart from the negligible collisions ($S_{i,k} = S_{j,k}$ for $i \neq j$), the reverse implication also holds. Then we can write

$$\Pr(\mathbf{A}_k = \mathbf{B}_k) = \Pr(\min(\mathbb{S}_A \cap \mathbb{S}_B) < \min((\mathbb{S}_A \cup \mathbb{S}_B) \setminus (\mathbb{S}_A \cap \mathbb{S}_B))) .$$

Let us now consider the right side of the above equation. Using the fact that minimum of exponentially distributed random variables is also exponentially distributed (see equation (1)) and that for $X \sim Exp(x)$, $Y \sim Exp(y)$ we have

$$\Pr(X < Y) = x/(x+y) \tag{10}$$

we can write

$$\Pr(\mathbf{A}_k = \mathbf{B}_k) = \frac{|\mathbb{A} \cap \mathbb{B}|_w}{|\mathbb{A} \cap \mathbb{B}|_w + |(\mathbb{A} \cup \mathbb{B}) \setminus (\mathbb{A} \cap \mathbb{B})|_w}$$

and thus

$$\Pr(\mathbf{A}_k = \mathbf{B}_k) = \frac{|\mathbb{A} \cap \mathbb{B}|_w}{|\mathbb{A} \cup \mathbb{B}|_w} \ . \tag{11}$$

□

Based on Lemma 4.1 and the Iverson bracket notation we can define the following estimator of $J_w(\mathbb{A}, \mathbb{B})$:

$$\bar{J}_w(\mathbf{A}, \mathbf{B}) := \frac{1}{m} \sum_{k=1}^{m} [\![\mathbf{A}_k = \mathbf{B}_k]\!] \ . \tag{12}$$

Note that the above sum represents the number of successes in m independent experiments, therefore it has binomial distribution with probability of success equal to $J_w(\mathbb{A}, \mathbb{B})$. Thus $\mathbb{E}(\bar{J}_w(\mathbf{A}, \mathbf{B})) = J_w(\mathbb{A}, \mathbb{B})$ and $\mathbb{V}\mathrm{ar}(\bar{J}_w(\mathbf{A}, \mathbf{B})) = J_w(\mathbb{A}, \mathbb{B})(1 - J_w(\mathbb{A}, \mathbb{B}))$. The definition of weighted Jaccard similarity presented in formula (9) can be generalized to any number of sets as:

$$J_w(\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots) := \frac{|\mathbb{A} \cap \mathbb{B} \cap \mathbb{C} \ldots|_w}{|\mathbb{A} \cup \mathbb{B} \cup \mathbb{C} \ldots|_w}. \tag{13}$$

Then it is easy to repeat the above reasoning to check that estimator

$$\bar{J}_w(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots) := \frac{1}{m} \sum_{k=1}^{m} [\![\mathbf{A}_k = \mathbf{B}_k = \mathbf{C}_k = \ldots]\!] \tag{14}$$

is an unbiased estimator of (13).

4.3 Intersection of sketches

Based on sketches **A** and **B** we can also define an unbiased estimator of the intersection $|\mathbb{A} \cap \mathbb{B}|_w$:

$$\bar{I}(\mathbf{A}, \mathbf{B}) := \bar{J}_w(\mathbf{A}, \mathbf{B})\ \bar{U}(\mathbf{A}, \mathbf{B})\ . \tag{15}$$

Namely, from (8) and (12) and linearity of expectation we get

$$\mathbb{E}\left(\bar{I}(\mathbf{A}, \mathbf{B})\right) = \frac{m-1}{m} \sum_{k=1}^{m} \mathbb{E}\left(\frac{[\![\mathbf{A}_k = \mathbf{B}_k]\!]}{\sum_{k=1}^{m} \min(\mathbf{A}_k, \mathbf{B}_k)}\right)\ . \tag{16}$$

Note that

$$\mathbb{E}\left(\frac{[\![\mathbf{A}_k = \mathbf{B}_k]\!]}{\sum_{k=1}^{m} \min(\mathbf{A}_k, \mathbf{B}_k)}\right) = \mathbb{E}\left(\frac{1}{\sum_{k=1}^{m} \min(\mathbf{A}_k, \mathbf{B}_k)} \;\middle|\; \mathbf{A}_k = \mathbf{B}_k\right) \Pr(\mathbf{A}_k = \mathbf{B}_k) \tag{17}$$

and by Lemma 4.2 proved below, we may drop the condition from the expression on the right side of equation (17). Moreover we may replace $\Pr(\mathbf{A}_k = \mathbf{B}_k)$ according to (11). In this way the right side of equation (17) becomes:

$$\mathbb{E}\left(\frac{1}{\sum_{k=1}^{m} \min(\mathbf{A}_k, \mathbf{B}_k)}\right) \cdot \frac{|\mathbb{A} \cap \mathbb{B}|_w}{|\mathbb{A} \cup \mathbb{B}|_w}\ . \tag{18}$$

By formula (8) we get

$$\mathbb{E}\left(\frac{1}{\sum_{k=1}^{m} \min(\mathbf{A}_k, \mathbf{B}_k)}\right) = \frac{\mathbb{E}(\bar{U}(\mathbf{A}, \mathbf{B}))}{m-1} = \frac{|\mathbb{A} \cup \mathbb{B}|_w}{m-1}\ . \tag{19}$$

and hence that $\bar{I}(\mathbf{A}, \mathbf{B})$ is unbiased estimator of $|\mathbb{A} \cap \mathbb{B}|_w$ for $m \geq 2$:

$$\mathbb{E}\left(\bar{I}(\mathbf{A}, \mathbf{B})\right) = \frac{m-1}{m} \sum_{k=1}^{m} \frac{|\mathbb{A} \cup \mathbb{B}|_w}{m-1} \frac{|\mathbb{A} \cap \mathbb{B}|_w}{|\mathbb{A} \cup \mathbb{B}|_w} = |\mathbb{A} \cap \mathbb{B}|_w\ . \tag{20}$$

Now it remains to prove the following lemma showing that the probability distribution of $\min(\mathbf{A}_k, \mathbf{B}_k)$ does not depend on whether or not equality $\mathbf{A}_k = \mathbf{B}_k$ holds.

Lemma 4.2. *For any $x \geq 0$ we have*

$$\Pr(\min(\mathbf{A}_k, \mathbf{B}_k) \leq x \mid \mathbf{A}_k = \mathbf{B}_k) = \Pr(\min(\mathbf{A}_k, \mathbf{B}_k) \leq x)\ .$$

Proof. Let us denote $p = |\mathbb{A} \cap \mathbb{B}|_w$ and $r = |(\mathbb{A} \cup \mathbb{B}) \setminus (\mathbb{A} \cap \mathbb{B})|_w$. Let us also define independent random variables $\mathbf{X}_1 \sim Exp(p)$ and $\mathbf{X}_2 \sim Exp(r)$. Then for $F_\alpha(x) = 1 - e^{-x\alpha}$ denoting cdf of the exponential distribution with parameter α we have

$$\Pr(\min(\mathbf{X}_1, \mathbf{X}_2) < x \wedge \mathbf{X}_1 < \mathbf{X}_2) = \Pr(\mathbf{X}_1 < \mathbf{X}_2 \wedge \mathbf{X}_1 < x) = \int_0^\infty \Pr(\mathbf{X}_1 < \min(z, x) \mid \mathbf{X}_2 = z)\, F_r'(z) dz = \int_0^\infty F_p(\min(z, x)) F_r'(z) dz = \frac{p}{p+r} F_{p+r}(x)\ . \tag{21}$$

Note that by the formula (10) we have $\Pr(\mathbf{X}_1 < \mathbf{X}_2) = p/(p+r)$. From this fact and equation (21) by using the definition of conditional probability and by noting that random variables $\min(\mathbf{X}_1, \mathbf{X}_2)$ and $\mathbf{X}_1 < \mathbf{X}_2$ are equivalent to $\min(\mathbf{A}_k, \mathbf{B}_k)$ and $\mathbf{A}_k = \mathbf{B}_k$ we get

$$\Pr(\min(\mathbf{A}_k, \mathbf{B}_k) \leq x \mid \mathbf{A}_k = \mathbf{B}_k) = F_{p+r}(x)\ .$$

We also know that

$$F_{p+r}(x) = 1 - e^{-x|\mathbb{A} \cup \mathbb{B}|_w} = \Pr(\min(\mathbf{A}_k, \mathbf{B}_k) \leq x)\ . \quad \square$$

Analogously to the expected value, with the use of Lemma 4.2, one can also derive the variance of estimator $\bar{I}(\mathbf{A}, \mathbf{B})$. For $m \geq 3$, $p = |\mathbb{A} \cap \mathbb{B}|_w$ and $s = |\mathbb{A} \cup \mathbb{B}|_w$ we get:

$$\mathbb{V}\text{ar}\left(\bar{I}(\mathbf{A}, \mathbf{B})\right) = \frac{p^2}{(m-2)m} + \frac{(m-1)ps}{(m-2)m} \approx \frac{ps}{m}\ . \tag{22}$$

By repeating the above reasoning one can check that the estimator defined analogously to (15) as

$$\bar{I}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots) := \bar{J}_w(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots)\, \bar{U}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots) \tag{23}$$

is an unbiased estimator of intersection $|\mathbb{A} \cap \mathbb{B} \cap \mathbb{C} \ldots|_w$.

The variance of $\bar{I}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots)$ can be expressed as in equation (22) with $p = |\mathbb{A} \cap \mathbb{B} \cap \mathbb{C} \ldots|_w$ and $s = |\mathbb{A} \cup \mathbb{B} \cup \mathbb{C} \ldots|_w$.

4.4 Partial sketch for intersection

One cannot expect that the sketches **A** and **B** could be stretched to a full-fledged sketch representing set $\mathbb{A} \cap \mathbb{B}$ since, by analogy to zooming in on an image, we lose resolution. However, we can create a partially defined sketch $\mathbf{A} \Cap \mathbf{B} = (\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_m)$ in which 0 represents an undefined position:

$$\mathbf{I}_k = \begin{cases} \mathbf{A}_k & \text{if } \mathbf{A}_k = \mathbf{B}_k\ , \\ 0 & \text{otherwise}\ . \end{cases}$$

Based on sketch $\mathbf{A} \Cap \mathbf{B}$ we can define an estimator $\hat{I}(\mathbf{A}, \mathbf{B})$ for $|\mathbb{A} \cap \mathbb{B}|_w$ analogously to the estimator $\bar{I}(\mathbf{A}, \mathbf{B})$ defined in the expression (15). Namely, we estimate $|\mathbb{A} \cup \mathbb{B}|_w$ by using formula (8) but taking into account only m' non-zero positions of sketch $\mathbf{A} \Cap \mathbf{B}$, since for these position we have $\mathbf{I}_k = \min(\mathbf{A}_k, \mathbf{B}_k)$. Jaccard similarity $J_w(\mathbb{A}, \mathbb{B})$ can be estimated by computing the ratio of the number of non-zero positions m' to the total number of positions in the sketch m. Since for non-zero positions we have $\mathbf{A}_k = \mathbf{B}_k$ we get exactly the same estimator as in formula (12). Therefore, by repeating the reasoning from equations (16) – (20) we may verify that

$$\hat{I}(\mathbf{A}, \mathbf{B}) := \frac{m'-1}{\sum_{\mathbf{I}_k \neq 0} \mathbf{I}_k} \cdot \frac{m'}{m} \tag{24}$$

is an unbiased estimator of $|\mathbb{A} \cap \mathbb{B}|_w$ for $m' \geq 2$.

Derivation of the variance of $\hat{I}(\mathbf{A}, \mathbf{B})$ seems to be more challenging since m' in contrast to m is a random variable. In Figure 3d in Section 4.7 we compare the variance of the estimators $\hat{I}$ and $\bar{I}$ experimentally. If $|\mathbb{A} \cap \mathbb{B}|_w$ is small and thus m' is much smaller than m, the variance of $\hat{I}$ is larger compared to variance of $\bar{I}$. Note however that the amount of information in the partial sketch is smaller compared to the full-fledged sketch.

On sketch $\mathbf{A} \Cap \mathbf{B}$ one can perform further operations with other sketches. For example, for sketch **C** one can create sketch $\mathbf{A} \Cap \mathbf{B} \Cap \mathbf{C}$ which at position k has

$$\mathbf{I}_k = \begin{cases} \mathbf{A}_k & \text{if } \mathbf{A}_k = \mathbf{B}_k = \mathbf{C}_k\ , \\ 0 & \text{otherwise}\ . \end{cases}$$

Based on sketch $\mathbf{A} \Cap \mathbf{B} \Cap \mathbf{C}$ one can use estimator $\hat{I}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ defined analogously to (24). Namely, let us note that from definition (13) we have $|\mathbb{A} \cap \mathbb{B} \cap \mathbb{C}|_w = |\mathbb{A} \cup \mathbb{B} \cup \mathbb{C}|_w \cdot J_w(\mathbb{A}, \mathbb{B}, \mathbb{C})$. One can estimate $|\mathbb{A} \cup \mathbb{B} \cup \mathbb{C}|_w$ by using the generalized union estimator $\bar{U}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ from Section 4.1 and m'' non-zero positions $\mathbf{I}_k = \min(\mathbf{A}_k, \mathbf{B}_k, \mathbf{C}_k)$ of sketch $\mathbf{A} \Cap \mathbf{B} \Cap \mathbf{C}$. Jaccard similarity $J_w(\mathbb{A}, \mathbb{B}, \mathbb{C})$ can be estimated based on definition (14) as the ratio m''/m, since for m'' non-zero positions of $\mathbf{A} \cap \mathbf{B} \cap \mathbf{C}$ we have $\mathbf{A}_k = \mathbf{B}_k = \mathbf{C}_k$.

The above reasoning can be easily generalized to a larger number of intersected sketches. Clearly, the number of non-zero positions in each successive intersection may decrease and thus the precision of estimates might deteriorate. Nonetheless, a partial sketch remains useful as long as it has at least two non-zero positions. An example discussed in Section 6 shows that constructing partial sketches based even on the large number of successive intersections does not necessarily cause a loss of precision.

4.5 Relative complement of sketches

Based on sketches $\mathbf{A}$ and $\mathbf{B}$ one can obtain an unbiased estimation for the relative complement $|\mathbb{A} \setminus \mathbb{B}|_w$ by using the intersection estimator (15) and the fact that

$$|\mathbb{A} \setminus \mathbb{B}|_w = |\mathbb{A}|_w - |\mathbb{A} \cap \mathbb{B}|_w \,. \tag{25}$$

We will show how to obtain a more precise estimation. Similarly as in the proof of Lemma 4.1 we can write

$$\Pr(\mathbf{A}_k < \mathbf{B}_k) = \Pr(\min(\mathbb{S}_A \setminus \mathbb{S}_B) < \min(\mathbb{S}_B)) = \frac{|\mathbb{A} \setminus \mathbb{B}|_w}{|\mathbb{A} \cup \mathbb{B}|_w} \,.$$

Therefore, by analogy to the Jaccard estimator (12), we can define an unbiased estimator for $\frac{|\mathbb{A} \setminus \mathbb{B}|_w}{|\mathbb{A} \cup \mathbb{B}|_w}$ as

$$\overline{CoU}(\mathbf{A}, \mathbf{B}) := \frac{1}{m} \sum_{k=1}^{m} [\![\mathbf{A}_k < \mathbf{B}_k]\!] \,. \tag{26}$$

Then, using reasoning analogous as in the formulas (16) - (20) and in the proof of Lemma 4.2 one can show that for $m \geq 2$

$$\overline{R}(\mathbf{A}, \mathbf{B}) = \overline{CoU}(\mathbf{A}, \mathbf{B})\, \overline{U}(\mathbf{A}, \mathbf{B}) \tag{27}$$

is unbiased estimator of $|\mathbb{A} \setminus \mathbb{B}|_w$. Moreover, the variance of $\overline{R}(\mathbf{A}, \mathbf{B})$ can be expressed with the variance formula (22) but with parameter $p = |\mathbb{A} \setminus \mathbb{B}|_w$. In Figure 3e in Section 4.7 we present experimental results comparing $\overline{R}(\mathbf{A}, \mathbf{B})$ with the naive estimator obtained with the formula (25). When the relative complement $|\mathbb{A} \setminus \mathbb{B}|_w$ is small, estimator $\overline{R}(\mathbf{A}, \mathbf{B})$ is noticeably more precise.

4.6 Partial sketch for relative complement

To construct a partial sketch representing $\mathbb{A} \setminus \mathbb{B}$ we can use a similar approach as for the intersection. Namely, we define a sketch $\mathbf{A} \setminus \mathbf{B} = (\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_m)$ where

$$\mathbf{R}_k = \begin{cases} \mathbf{A}_k & \text{if } \mathbf{A}_k = \mathbf{B}_k \,, \\ < & \text{if } \mathbf{A}_k < \mathbf{B}_k \,, \\ > & \text{if } \mathbf{A}_k > \mathbf{B}_k \,. \end{cases}$$

Then we can estimate $|\mathbb{A} \cup \mathbb{B}|_w$ by using formula (8) with elements of the sketch $\mathbf{A} \setminus \mathbf{B}$ different than the symbols $<, >$, since for these elements $\mathbf{R}_k = \min(\mathbf{A}_k, \mathbf{B}_k)$. To get the estimation for the complement over union $\overline{CoU}(\mathbf{A}, \mathbf{B})$ as in the formula (26) we find the ratio of the number of $<$ symbols in the sketch $\mathbf{A} \setminus \mathbf{B}$ to the length of sketch m. By taking the product of this two estimations in the same way as in case of the estimator $\hat{I}(\mathbf{A}, \mathbf{B})$ we obtain an unbiased estimator $\hat{R}(\mathbf{A}, \mathbf{B})$ of $|\mathbb{A} \setminus \mathbb{B}|_w$. However, as shown in the Figure 3f in Section 4.7, when the relative complement $|\mathbb{A} \setminus \mathbb{B}|_w$ is large estimator $\hat{R}(\mathbf{A}, \mathbf{B})$ is significantly less precise than $\overline{R}(\mathbf{A}, \mathbf{B})$.

4.7 Experimental verification

In this section we experimentally verify analytical results from sections 4.1 – 4.6. Figures 3a – 3f present the results of simulation on sets

$$\mathbb{A} = \{(i, 1) : i = 1, 2, \ldots, n\} \,,$$
$$\mathbb{B}_j = \{(j + i, 1) : i = 1, 2, \ldots, n\} \tag{28}$$

for $n = 1000$. With these sets we have the clear situation where $|\mathbb{A} \cup \mathbb{B}_j|_w = n + j$, $|\mathbb{A} \cap \mathbb{B}_j|_w = n - j$, $|\mathbb{A} \setminus \mathbb{B}_j|_w = j$. Further we will use these three formulas to normalize the values of estimators $\overline{U}(\mathbf{A}, \mathbf{B}_j)$, $\overline{I}(\mathbf{A}, \mathbf{B}_j)$, $\overline{R}(\mathbf{A}, \mathbf{B}_j)$, respectively, to have them close to 1.

In Figure 3a we present a histogram of results returned for set $\mathbb{A}$ by the sum estimator $\overline{\Lambda}$ defined in (4) for $m = 200$. From our experiments it follows that distribution of $\overline{\Lambda}$ becomes similar to the normal distribution as value of parameter m increases. Note that from the central limit theorem a random variable with the gamma distribution as the sum of m independent exponentially distributed random variables can be well approximated by the normal distribution for sufficiently large m.

In Figure 3b we present the value of $\overline{\Lambda}/\Lambda$ for set $\mathbb{A}$ and m changing from 2 to 1000. Since we know the variance of $\overline{\Lambda}$ (equation (5)) we can also plot six functions relating to the three-sigma rule. Namely, for normally distributed random variable approximately 68.27%, 95.45% and 99.73% of the values lie within one, two and three standard deviations of the mean, respectively. In our case the six functions are defined as

$$1 \pm \sigma[\overline{\Lambda}/\Lambda],\ 1 \pm 2\sigma[\overline{\Lambda}/\Lambda],\ 1 \pm 3\sigma[\overline{\Lambda}/\Lambda]$$

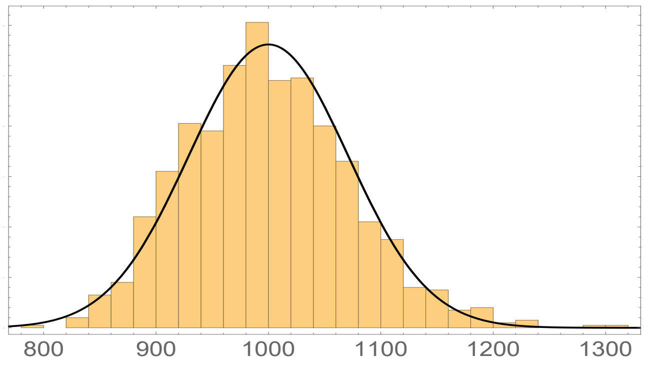
where $\sigma[X] = \sqrt{\mathbb{V}\mathrm{ar}\,(X)}$ denotes the standard deviation of X.

In Figures 3c–3f we set $m = 200$ and consider operations on sets $\mathbb{A}$ and $\mathbb{B}_j$ as j changes from 1 to n. In Figure 3c we present normalized values of the union estimator $\overline{U}(\mathbf{A}, \mathbf{B}_j)$. Note that $\overline{U}$ is in fact the sum estimator $\overline{\Lambda}$ on set $\mathbb{A} \cup \mathbb{B}_j$. Thus for a fixed m the concentration of the normalized estimator $\overline{U}$ does not change as j changes. Moreover, by using the variance of $\overline{\Lambda}$ from equation (5) we can also draw six functions corresponding to the three-sigma rule. Note that in this case for fixed m these functions are constant.

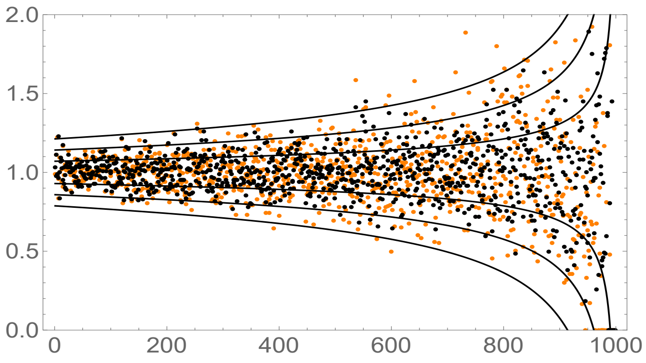
Figure 3d presents normalized values of intersection estimator $\overline{I}(\mathbf{A}, \mathbf{B}_j)$ (black dots) and intersection estimator based on the partial sketch $\hat{I}(\mathbf{A}, \mathbf{B}_j)$ (orange dots). One can observe that the precision of $\hat{I}$ deteriorates compared to $\overline{I}$ as the size of the estimated intersection $|\mathbb{A} \cap \mathbb{B}_j|_w = n - j$ goes to zero. Using the variance formula (22) we also draw six functions related to the three-sigma rule for $\overline{I}(\mathbf{A}, \mathbf{B}_j)$.

In Figure 3e we compare normalized values of the relative complement estimator $\overline{R}(\mathbf{A}, \mathbf{B}_j)$ (black dots) to the normalized naive estimator obtained from the formula (25) (orange dots). One can observe that the precision of the naive estimator clearly deteriorates compared to $\overline{R}$ as the estimated relative complement $|\mathbb{A} \setminus \mathbb{B}_j|_w = j$ goes to zero. In Figure 3f we compare normalized values of estimator $\overline{R}(\mathbf{A}, \mathbf{B}_j)$ (black dots) to normalized values of estimator $\hat{R}(\mathbf{A}, \mathbf{B}_j)$ based on the partial sketch (orange dots). Precision of $\hat{R}$ clearly deteriorates compared to $\overline{R}$ as the value of $|\mathbb{A} \setminus \mathbb{B}_j|_w = j$ increases.

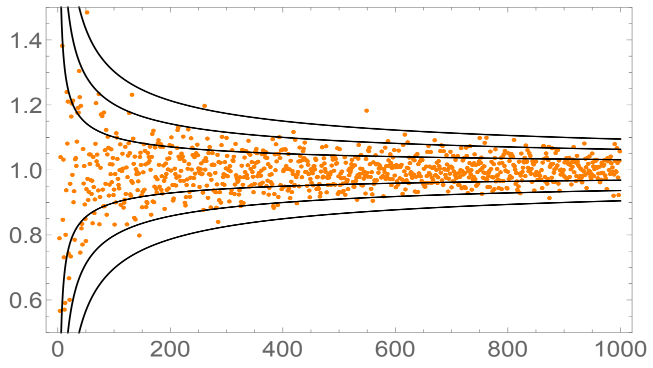
In Figures 3e and 3f, based on the variance formula (22) with $p = |\mathbb{A} \setminus \mathbb{B}_j|_w$ and $s = |\mathbb{A} \cup \mathbb{B}_j|_w$ we also draw six functions corresponding to the three-sigma rule for estimator $\overline{R}$.



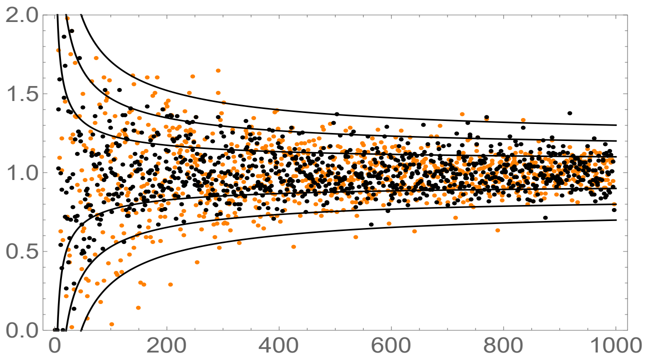
(a) Histogram of 1000 independent realizations of estimator $\overline{\Lambda}$ with parameter $m = 200$ on set $\mathbb{A}$. Black curve represents the normal distribution with mean $\mathbb{E}(\overline{\Lambda})$ and variance $\mathbb{V}\mathrm{ar}(\overline{\Lambda})$.



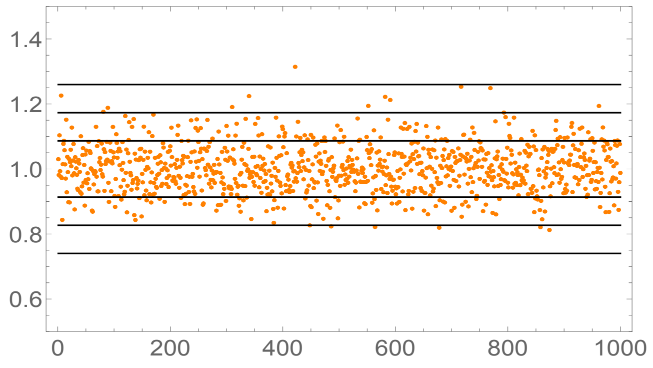
(d) Black dots represent normalized values of $\overline{I}(\mathrm{A}, \mathrm{B}_j)$, orange normalized values of $\hat{I}(\mathrm{A}, \mathrm{B}_j)$ for $m = 200$ and $j \in \{1, 2, \ldots, 1000\}$. Black lines show the three-sigma rule for $\overline{I}$ obtained with equation (22).



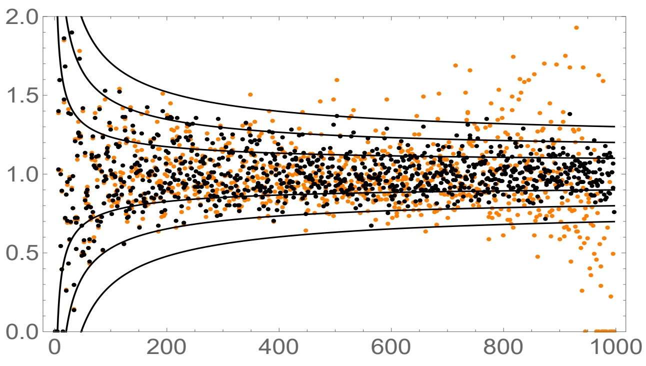
(b) Orange dots represent values of $\overline{\Lambda}/\Lambda$ for the set $\mathbb{A}$ as parameter m changes from 2 to 1000. Six black lines represent bounds related to the three-sigma rule obtained with equation (5) . The concentration of $\overline{\Lambda}/\Lambda$ increases with m and does not depend on the set $\mathbb{A}$.



(e) Black dots represent normalized values of $\overline{R}(\mathrm{A}, \mathrm{B}_j)$ and orange dots normalized naive estimation based on equation (25) for $m = 200$ and $j \in \{1, 2, \ldots, 1000\}$. Black lines represent the three-sigma rule for $\overline{R}$ based on equation (22) with $p = |\mathbb{A} \setminus \mathbb{B}_j|_w$ and $s = |\mathbb{A} \cup \mathbb{B}_j|_w$.



(c) Orange dots represent normalized values of $\overline{U}(\mathrm{A}, \mathrm{B}_j)$ for $m = 200$ and $j \in \{1, 2, \ldots, 1000\}$. Using $\overline{U}(\mathrm{A}, \mathrm{B}_j)$ is equivalent to using estimator $\overline{\Lambda}$ on set $\mathbb{A} \cup \mathbb{B}_j$. Thus for a fixed m the precision of the normalized estimation does not change with j. Six black lines represent bounds related to the three-sigma rule obtained with equation (5) .



(f) Black dots represent normalized values of $\overline{R}(\mathrm{A}, \mathrm{B}_j)$, orange normalized values of $\hat{R}(\mathrm{A}, \mathrm{B}_j)$ for $m = 200$ and $j \in \{1, 2, \ldots, 1000\}$. Precision of $\hat{R}$ deteriorates compared to $\overline{R}$ as the value of $|\mathbb{A} \setminus \mathbb{B}_j|_w = j$ increases. Black lines represent the three-sigma rule for $\overline{R}$ obtained with equation (22) with $p = |\mathbb{A} \setminus \mathbb{B}_j|_w$ and $s = |\mathbb{A} \cup \mathbb{B}_j|_w$.

Figure 3: Figures 3a - 3f present simulation results for different operators on sets $\mathbb{A}$ and $\mathbb{B}_j$ defined in the formula (28).

5 APPLICATIONS

The natural application for the proposed framework is building data aggregation scheme for many independently working data collection entities (e.g. sensors, crawlers, routers, satellites). In such applications the framework not only enables effective collecting and aggregation procedures but also flexible manipulation on the aggregated data. The framework can also be used in the construction of distributed databases. Here we describe a few selected scenarios.

5.1 Opportunistic data collection

Let us concern the following problem: there are multiple data streams $S_1, \ldots, S_t$ and the agents – so called *mules* – $A_1, \ldots, A_m$ collecting data. For example, $S_1, \ldots, S_t$ represent sensors in a sensor field, while $A_1, \ldots, A_m$ are mobile users collecting the data measured by $S_1, \ldots, S_t$ (for model details see e.g. [35]). The goal is to collect the measurements and to compute, say, the average value. Obviously, this can be achieved if the lists of original measurements together with unique identifiers are provided to the sink – a data collection point. However, if the data are aggregated by the agents, then a careful coordination is necessary so that no value is counted twice (e.g. dividing the sensor field into subregions and assigning exactly one agent to one subregion).

Data sketches enable running the data collection process in opportunistic way without any central coordination. The mobile mules $A_1, \ldots, A_m$ simply create independently their data sketches and from time to time deliver them to the sink. The sink aggregates the sketches (e.g. by the union operation) and thereby learns the result for stream data received collectively by all mules. Note that the scheme is tolerant to mules entering or leaving the system. It is also tolerant to malicious mules attempting to manipulate the result. If the malicious goal is to reduce the total, then nothing can prevent counting a value read by an honest mule. If the malicious goal is to increase the total, then it is easy the prevent the attack. Namely, each data stream element is authenticated by its source (e.g. by means of a digital signature), and this authentication string is attached to $M[j, k]$ if the operation from line 5 of CreateSketch algorithm is executed. In this case the new authentication string overwrites the previous one attached to $M[j, k]$.

5.2 Data aggregation in wireless networks

Again, we are talking about collecting the data generated by network nodes $S_1, \ldots, S_t$ and delivering the totals to the sink. However now we assume that the nodes $S_1, \ldots, S_t$ create communication network themselves. There are many strategies how to organize the communication network (see e.g. the survey [36]). The major problems in this area is to protect the network from communication errors or adversary trying to prevent delivery of some measurements to the sink. If the adversary can either corrupt some nodes or interrupt some connections, then the only defense is to send the same data over multiple paths. However, if the data has to be aggregated by the intermediate nodes, then the problem of double counting arises.

Data sketches solve this problem in an elegant way: a network node aggregates received data sketches using methods from Section 4 and forwards the resulting sketch. So even if the network follows a flooding strategy, this does not result in an explosion of the data size while certain accuracy of the data finally delivered is guaranteed (see: extrema propagation technique [3, 17]). An additional advantage of this approach is that it is independent of the network structure, so reconfiguration of routing paths can be done without interrupting data collection.

5.3 Queries to distributed data sets

Due to many reasons (e.g. privacy protection according to GDPR) data might be distributed over multiple databases, while databases may contain the same entries. This may occur for medical databases, where the same entry concerning a particular patient may be duplicated to the medical databases of the doctors responsible for a treatment of this patient.

The problem arises when a query concerns data contained in all databases that are not disjoint. For privacy protection reasons each database should return only aggregated results (e.g. the number of cases of a particular medical event). If the raw numbers are released, then it is impossible to avoid double counting. The correct result would be generated, if a `join` operation is performed prior to evaluating the query, but this would be inefficient and possibly illegal from the point of view of GDPR. Data sketches solve this problem in an elegant and efficient way: each database releases a data sketch as an answer and it suffices to aggregate the sketches using operations described in Section 4.

5.4 Privacy

There are different methods to reduce the leakage of private information when releasing the aggregates. For instance, one of popular methods is to add noise to a private data that is an input to the data aggregation procedure (see e.g. [11]). Data sketches offer an alternative mechanism: instead of providing the data directly, a given entry may or may not be used for an update in line 5 of the CreateSketch algorithm. So most of the entries have not explicit impact on the final shape of the sketch. For example, one may think about the traffic flow analysis: to analyze the flow we can aggregate sketches from different locations and time windows, but we won't be able to track the movement of a single car.

Moreover, the operations over the sketch can be performed on encrypted data and as a multi-party protocol. All we have to do is to implement the operations from the step 5 of CreateSketch algorithm on ciphertexts. If the hash values can be generated on the source side then it is enough to change the definition of u in line 4 of the CreateSketch algorithm to

$$u \leftarrow h\left(i^\frown j^\frown k^\frown \lambda_{i,j}\right) \tag{29}$$

since now ciphertext $c = -\ln u / \lambda_{i,j}$ effectively hides value of $\lambda_{i,j}$. If only the ciphertext of $\lambda_{i,j}$ is to be send to the aggregator there are two crucial issues to be solved: how to compare the current value of $M[j, k]$ with $-\ln u / \lambda_{i,j}$ and how to show that u has been created correctly. At this point note that from security point of view it would be desirable to change the definition of u such that it would take into account the value of $\lambda_{i,j}$ similarly as in the formula (29).

6 EXPERIMENTS

To verify the framework we model a scenario based on the actual problem in a multi-hop wireless sensor network we came across.

In such networks communication errors as well as node failures are common, so sending a message via a single path could be risky. A popular strategy is to forward a message to some number of neighbors that are closer to the destination. We consider a special case of this approach known as the braid chain strategy (see [16]).

Braid chain. A braid chain of length L consists of two paths, each of length L. Nodes that are on the same position in two paths are considered to be in the same layer, so we have two nodes per layer, say A_i and B_i in the layer i. Each node from layer i can send a message to both nodes of layer $i+1$. Let us denote the probability of a successful transmission between two nodes on the same path by p and probability of a successful transmission between two nodes on different paths by q (see Figure 4). In [16] it was shown the probability that communication in the braid chain is not disconnected is much higher to the analogical probability for two separate paths. In further experiments we set $p = 0.9$, $q = 0.1$ and $L = 30$. Note however that p and q do not have to sum up to 1.

Data sources. We will use data sketches to analyze the packet traffic in the braid chain. As depicted in Figure 4 we assume there are two independent data sources $\mathbb{A}$, $\mathbb{B}$ and the sink $\mathbb{S}$. The size of packets generated by $\mathbb{A}$ and $\mathbb{B}$ is described by random variables X and $X+1$, respectively, where X follows the beta distribution with parameters $\alpha = \beta = 5$ (see Figure 5a). The beta distribution is similar to the normal distribution but it does not have long tails – it is defined on $[0,1]$ interval. Thus, the beta distribution provides more realistic simulations of a packet size, that match the practical results (see [8]). This can be justified by the limited range of a packet size in the standard transport protocols. We assume that each of two sources generates $n = 10^4$ packets.

Mean and variance. To estimate the mean and variance of size of packets that go through each node of the braid chain we use the trick from Section 3.3. Namely, we encode each packet as a tuple with three features $(i, 1, \lambda_i, \lambda_i^2)$, where i is a unique identifier of a packet and λ_i represents its size. In each node of the braid chain we create a sketch with $d = 3$ rows and $m = 200$ columns. Let us name sketches created in nodes A_i and B_i as $\mathbf{A}_i$ and $\mathbf{B}_i$, respectively. Recall that the first row of each sketch can be used to estimate the number of seen packets n, the second row to estimate their total size $\sum_{i=1}^{n} \lambda_i$ and third row to estimate $\sum_{i=1}^{n} (\lambda_i)^2$.

Based on the procedure described in Section 3.3 in each node we create a sketch and estimate the mean and variance of size of packets that have passed through this node. The results for nodes on path A of the braid chain are presented in Figure 5b. In this and all subsequent figures the exact results are represented by solid lines

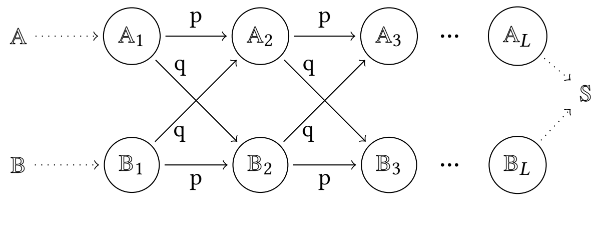


Figure 4: Braid chain of length L with sources $\mathbb{A}$, $\mathbb{B}$ and sink $\mathbb{S}$. $\mathbb{A}_i$ represents a set of packets seen by node A_i.

and results obtained with sketches by dashed lines. As expected for node A_1 the mean size of packets is close to $\mathbb{E}(X) = 1/2$. In subsequent nodes, as packets from sources $\mathbb{A}$ and $\mathbb{B}$ get mixed, the mean size grows to 1, which is the mean size of packets in set $\mathbb{A} \cup \mathbb{B}$.

Note that as mentioned in Section 3.3 estimates for the variance obtained by the standard formula – represented in Figure 5b by the black dashed line – are not very stable. Therefor, we have applied Welford's online algorithm for estimating the variance based on the sketches (see Section 3.3). The results obtained by Welford's algorithm are represented in Figure 5b by the black dotted line and, as expected, approximate the true variance much better.

Lost packets. Sketches $\mathbf{A}_i$ and $\mathbf{B}_i$ described above can be also used to estimate the number and the total size of packets lost in consecutive layers of the braid chain. For example, the set of packets from source $\mathbb{A}$ lost up to layer i can be expressed as

$$\mathbb{L}_i^{\mathbb{A}} := \mathbb{A}_1 \setminus (\mathbb{A}_i \cup \mathbb{B}_i) .$$

Thus, the number of lost packets $|\mathbb{L}_i^{\mathbb{A}}|$ and their total size $|\mathbb{L}_i^{\mathbb{A}}|_w$ can be estimated based on union and relative complement operations on sketches $\mathbf{A}_1, \mathbf{A}_i, \mathbf{B}_i$. More precisely, to estimate $|\mathbb{L}_i^{\mathbb{A}}|$ and $|\mathbb{L}_i^{\mathbb{A}}|_w$ we should use the first and second row of these sketches, respectively.

The results of estimations for $|\mathbb{L}_i^{\mathbb{A}}|$ and $|\mathbb{L}_i^{\mathbb{A}}|_w$ for each layer i are presented in Figure 5c. Since for the source $\mathbb{A}$ the mean size of a packet is $1/2$, therefore on average the total size of packets is two times smaller than their number. Note that with the chosen parameters $p = 0.9$ and $q = 0.1$ up to the 10th layer we have lost approximately 50% from $n = 10^4$ packets. Note also that on a single path with parameter $p = 0.9$ up to the 10th layer we would lose on average $1 - (0.9)^9 \approx 61\%$ of packets.

Packet flow analysis. Using sketches $\mathbf{A}_i$ and $\mathbf{B}_i$ we can also analyze the presence of packets of a given type in selected subsets of network nodes. For example, we may study the total size of packets from source $\mathbb{A}$ or source $\mathbb{B}$ that were present in consecutive nodes $i \in \{1, 2, \ldots, L\}$ of path A. Namely, for sets

$$A_i^{\mathbb{A}} := \mathbb{A}_1 \cap \mathbb{A}_i \ , \quad A_i^{\mathbb{B}} := \mathbb{B}_1 \cap \mathbb{A}_i \tag{30}$$

we can estimate $|A_i^{\mathbb{A}}|_w$ and $|A_i^{\mathbb{B}}|_w$. Similarly, we may be interested in the total size of packets from both sources that were present in the consecutive nodes of path A:

$$|\,(\mathbb{A}_1 \cup \mathbb{B}_1) \cap \mathbb{A}_i|_w \ .$$

We can also investigate the total size of packets from source $\mathbb{A}$ that were always present up to a given node i on nodes of path A:

$$|\mathbb{A}_1 \cap \mathbb{A}_2 \cap \ldots \cap \mathbb{A}_i|_w \tag{31}$$

or the total size of packets from source $\mathbb{B}$ that were ever present up to a given node i on nodes of path A:

$$|\mathbb{B}_1 \cap (\mathbb{A}_1 \cup \mathbb{A}_2 \cup \ldots \cup \mathbb{A}_i)\,|_w \ .$$

The experimental results for the above weighted cardinalities as value of i changes from 1 to L are presented in Figures 5d and 5e. In Figure 5d we see that the total size of packets from source $\mathbb{A}$ on path A starts at 5000 in node A_1 and drops to approximately 500 in node A_{30}. The total size of packets from source $\mathbb{B}$ starts at 0 in node A_1, grows to approximately 4500 in node A_{10} and starts to drop. As expected, the total size of packets from both sources is the sum of the total sizes of packets of each type. In Figure 5e we see that for node A_{30} the total size of packets from source $\mathbb{A}$ that were always present on path A is approximately equal to 250. Since the total size of packets from source $\mathbb{A}$ in node A_{30} is 500 (see Figure 5d) it means the other 250 was restored due to the presence of path B.

Partial sketches. To estimate the weighted cardinality defined in formula (31) one can collect all sketches $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_i$ in one place and perform the operation defined in formula (23). However, such approach entails a considerable communication overhead. An alternative solution is to use partial sketches. Namely, node A_1 sends sketch $\mathbf{A}_1$ to node A_2, which creates partial sketch $\mathbf{A}_1 \Cap \mathbf{A}_2$ and sends it to node A_3 which creates sketch $\mathbf{A}_1 \Cap \mathbf{A}_2 \Cap \mathbf{A}_3$, etc. In general node A_i creates partial sketch $\mathbf{A}_1 \Cap \mathbf{A}_2 \Cap \ldots \Cap \mathbf{A}_i$ on which estimator $\hat{I}(\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_i)$ described in Section 4.4 can be used to estimate the value of (31). The results returned by estimator $\hat{I}$ for successive values of i are shown in Figure 5e as black dots.

Nodes similarity. As packets from sources $\mathbb{A}$ and $\mathbb{B}$ are getting mixed in the braid chain, sets $\mathbb{A}_i$ and $\mathbb{B}_i$ in consecutive layers become similar to each other. We can determine how similar they are by using the weighted Jaccard similarity $J_w(\mathbb{A}_i, \mathbb{B}_i)$. However, in this way we do not consider the source of packets, but only what percentage of the total size of packets in a layer is common for both nodes. To take into account the source of packets we may use the total variation distance

$$||\mathbf{a} - \mathbf{b}|| := \frac{1}{2} \sum_{y \in Y} |\mathbf{a}(y) - \mathbf{b}(y)|$$

representing the similarity of two probability distributions $\mathbf{a}$, $\mathbf{b}$ on the countable space Y as a number between 0 and 1 (see e.g. [31]). Namely, using the notation defined in (30), we can compute what fraction of the total size of packets at node A_i comes from source $\mathbb{A}$

$$p^{\mathbb{A}}_{A_i} := \frac{|A^{\mathbb{A}}_i|_w}{|A^{\mathbb{A}}_i|_w + |A^{\mathbb{B}}_i|_w}$$

and analogically find probability distributions for nodes A_i and B_i

$$\mathbf{p}_{A_i} := \left(p^{\mathbb{A}}_{A_i}, p^{\mathbb{B}}_{A_i}\right) \ , \quad \mathbf{p}_{B_i} := \left(p^{\mathbb{A}}_{B_i}, p^{\mathbb{B}}_{B_i}\right) \ .$$

Therefore, we can compute the total variation distance $||\mathbf{p}_{A_i} - \mathbf{p}_{B_i}||$ to measure how similar are sets $\mathbb{A}_i$ and $\mathbb{B}_i$ in terms of where their elements come from.

Simulations for Jaccard similarity $J_w(\mathbb{A}_i, \mathbb{B}_i)$ and total variation distance $||\mathbf{p}_{A_i} - \mathbf{p}_{B_i}||$ for $i \in \{1, \ldots, 30\}$ are presented in Figure 5f. $J_w(\mathbb{A}_i, \mathbb{B}_i)$ starts at value 0 for $i = 1$ and grows to 0.45 for $i = 30$. $||\mathbf{p}_{A_i} - \mathbf{p}_{B_i}||$ starts at value 1 for $i = 1$ and drops to 0 for $i = 30$, which means that the proportion of total size of packets from source $\mathbb{A}$ and $\mathbb{B}$ is roughly the same for nodes A_{30} and B_{30}.

7 PRACTICAL NOTES AND CONCLUSIONS

Memory usage. The most obvious benefit of employing sketches compared to storing raw data is low memory usage. Storing a sketch requires $d \cdot m \cdot b$ bits, where $d \cdot m$ is the sketch size and b is the number of bits per hash. Parameters d and m were discussed in Section 3 and do not depend on the number of elements in the input stream. The optimal value of the parameter m depends on the precision required in a given use case, however from Figure 3b it can be concluded that $m = 200$ seems to be a reasonable default value. The number of bits per hash b should be chosen so that the probability of hash collisions for different elements is low. If the number of collisions is substantial compared to number of elements, the value obtained in line 5 of Algorithm 1 can be higher than it should be and the results based on a sketch might be underestimated. Assuming a hash function provides a uniform distribution the Birthday Paradox guarantees that the probability of a collision is low if the number of hashed elements n do not exceed $\sqrt{2^b}$. For example, using $b = 64$ bits per hash is safe up to approximately $\sqrt{2^{64}} \approx 4 \cdot 10^9$ elements. Transformation $f(u) = -\ln(u)/\lambda$ applied in the algorithm to hash values is a one-to-one expansion mapping from the interval $[0, 1)$ onto $\mathbb{R}_+$ so it should not cause collisions. However, the use of an inefficient method for computing the value of the logarithm may affect the performance of the algorithm (see e.g. [30]).

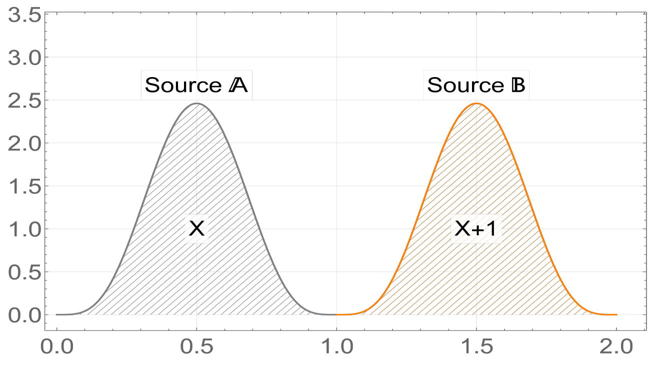
Note that for unweighted sets the memory-precision trade-off is almost exactly the same for our solution and for KMV sketches.

Finally, lest us remark that as the number of elements increases the values stored in the sketch decrease, so a compression algorithm based on removing unused leading bits could be proposed.

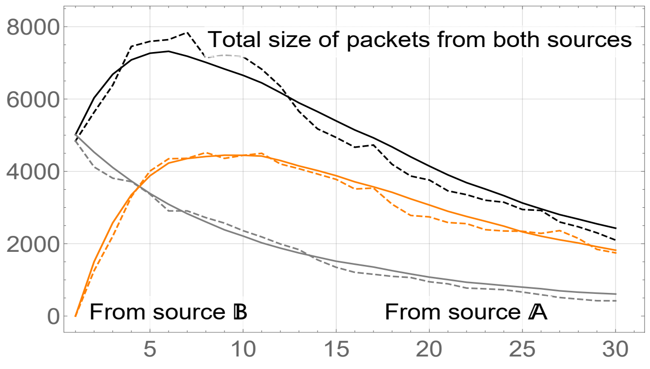
Hash function. The key to the algorithm's performance is the choice of a hash function. An appropriate hash function should (1) be fast enough, (2) map data uniformly to the range, (3) resemble a truly random hash function – namely, map each data item independently. In applications having big-size input data and requiring fast processing, for which security aspects are not crucial, non-cryptographic hash functions like MurMurHash3 and Fowler–Noll–Vo are currently a popular choice (see [12]). In our experiments on the braid chain described in Section 6 we relied on the MurMurHash3 function and found no disturbing signals. It seems that even for very massive data streams these hash functions should meet postulates (1) and (2) although it is not entirely clear whether they will provide adequate independence of experiments. Fortunately, in practice even simple hash functions are commonly observed to perform as predicted by the idealized analysis for truly random hash functions. This phenomenon arises naturally from a combination of the randomness of the hash function and randomness present in the data (see [15]). Note also that in Algorithm 1 each experiment is based on computing minimum of hashed values, so an ideal solution would be to construct a family of hashing functions providing min-wise independence analogical to this defined in [7] but additionally taking into account the weights. Finally, let us remark that if we apply the approach proposed in Section 3.4 the experiments corresponding to different rows of the sketch do not have to be independent.

Conclusions. In the paper we generalize the problem of evaluating set theory operations based on data sketches to the weighted sets. We analyze the solution that relies heavily on the properties of the exponential distribution. As for the practical issues, more research is needed in selecting a hashing scheme. In particular one could try to look for family of functions providing min-wise independence as this defined in [7] but taking into account the weights.

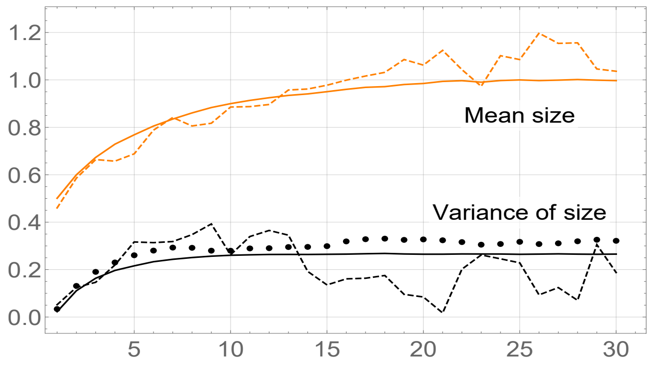
It also seems that it is possible to incorporate mechanisms related to cryptography into the sketch to provide data privacy or, conversely, data accountability by relating it to the submitting entity. The latter one can ensure the integrity of the sketch (cf. Section 5.1). Further research is required to refine the solution in this respect.



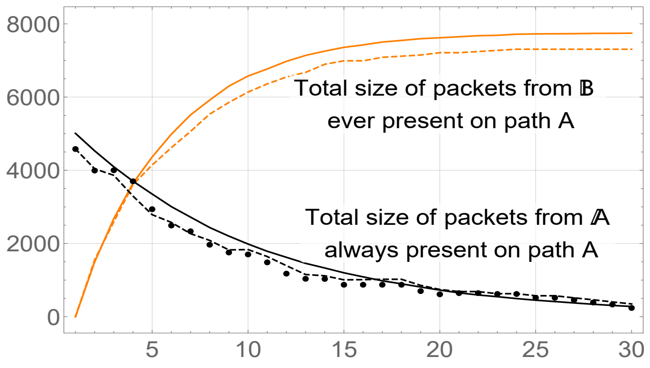
(a) Probability distributions for the random variables X and $X + 1$. Variable X follows the beta distribution with parameters $\alpha = \beta = 5$.



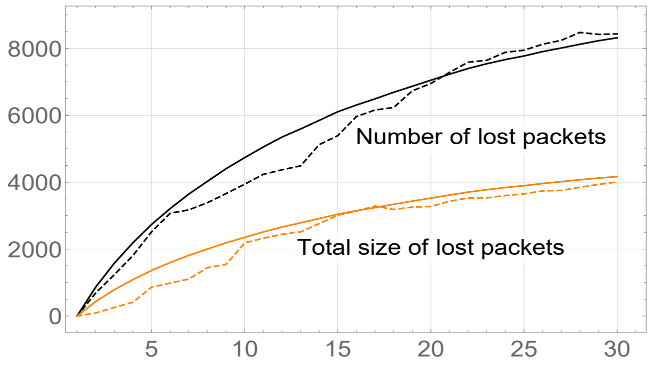
(d) The total size of packets from source $\mathbb{A}$, from source $\mathbb{B}$ and from both sources that were present in consecutive nodes of path A.



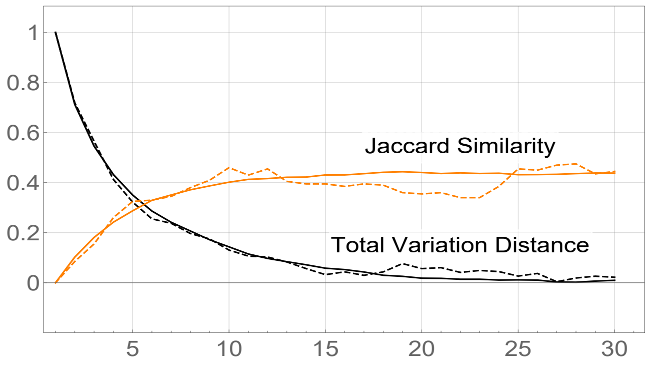
(b) The mean and variance of size of packets passing through each node A_1 to A_{30}. The black dots represent estimates of the variance obtained with the Welford's online algorithm based on sketches.



(e) The total size of packets from source $\mathbb{A}$ always present and from source $\mathbb{B}$ ever present in consecutive nodes of path A. The black dots represent estimates obtained based on partial sketches.



(c) The number $|\mathbb{L}_i^{\mathbb{A}}|$ and the total size $|\mathbb{L}_i^{\mathbb{A}}|_w$ of packets from source $\mathbb{A}$ that have been lost up to the layer i for $i \in \{1, \ldots, 30\}$.



(f) The Jaccard similarity $J_w(\mathbb{A}_i, \mathbb{B}_i)$ and the total variation distance $||p_{A_i} - p_{B_i}||$ for $i \in \{1, \ldots, 30\}$.

Figure 5: The figure presents experimental results for the braid chain of length $L = 30$ with parameters $p = 0.9$ and $q = 0.1$. In Figures 5b - 5f solid lines represent exact results, dashed lines represent results based on sketches with parameter $m = 200$.

REFERENCES

[1] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff Phillips, Zhewei Wei, and Ke Yi. 2012. Mergeable Summaries. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems* (Scottsdale, Arizona, USA) *(PODS '12)*. Association for Computing Machinery, New York, NY, USA, 23–34. https://doi.org/10.1145/2213556.2213562

[2] Noga Alon, Yossi Matias, and Mario Szegedy. 1996. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (Philadelphia, Pennsylvania, United States) *(STOC '96)*. ACM, New York, NY, USA, 20–29.

[3] Carlos Baquero, Paulo Sérgio Almeida, and Raquel Menezes. 2009. Fast Estimation of Aggregates in Unstructured Networks. In *Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems*. IEEE Computer Society, 88–93.

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. 2002. Counting Distinct Elements in a Data Stream. In *RANDOM*. 1–10.

[5] Kevin Beyer, Rainer Gemulla, Peter J. Haas, Berthold Reinwald, and Yannis Sismanis. 2009. Distinct-Value Synopses for Multiset Operations. *Commun. ACM* 52, 10 (Oct. 2009), 87–95. https://doi.org/10.1145/1562764.1562787

[6] Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*. IEEE Computer Society, 21–29. https://doi.org/10.1109/SEQUEN.1997.666900

[7] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. 1998. Min-Wise Independent Permutations (Extended Abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (Dallas, Texas, USA) *(STOC '98)*. Association for Computing Machinery, New York, NY, USA, 327–336. https://doi.org/10.1145/276698.276781

[8] Ewerton RS Castro, Marcelo S Alencar, and Iguatemi E Fonseca. 2013. Probability density functions of the packet length for computer networks with bimodal traffic. *International Journal of Computer Networks & Communications* 5, 3 (2013), 17.

[9] Tony F. Chan, Gene H. Golub, and Randall J. Leveque. 1983. Algorithms for Computing the Sample Variance: Analysis and Recommendations. *The American Statistician* 37, 3 (1983), 242–247. https://doi.org/10.1080/00031305.1983.10483115

[10] Philippe Chassaing and Lucas Gerin. 2006. Efficient estimation of the cardinality of large data sets. In *4th Colloquium on Mathematics and Computer Science*. DMTCS Proceedings, 419–422.

[11] Yuwen Chen, José-Fernán Martínez, Pedro Castillejo, and Lourdes López. 2018. A Privacy-Preserving Noise Addition Data Aggregation Scheme for Smart Grid. *Energies* 11, 11 (Nov 2018), 2972. https://doi.org/10.3390/en11112972

[12] Lianhua Chi and Xingquan Zhu. 2017. Hashing Techniques: A Survey and Taxonomy. *ACM Comput. Surv.* 50, 1, Article 11 (April 2017), 36 pages. https://doi.org/10.1145/3047307

[13] Flavio Chierichetti, Ravi Kumar, Sandeep Pandey, and Sergei Vassilvitskii. 2010. Finding the Jaccard Median. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms* (Austin, Texas) *(SODA '10)*. Society for Industrial and Applied Mathematics, USA, 293–311.

[14] Ondrej Chum, James Philbin, and Andrew Zisserman. 2008. Near Duplicate Image Detection: min-Hash and tf-idf Weighting.. In *BMVC*, Mark Everingham, Chris J. Needham, and Roberto Fraile (Eds.). British Machine Vision Association, 1–10. http://dblp.uni-trier.de/db/conf/bmvc/bmvc2008.html#ChumPZ08

[15] Kai-Min Chung, Michael Mitzenmacher, and Salil Vadhan. 2013. Why Simple Hash Functions Work: Exploiting the Entropy in a Data Stream. *Theory of Computing* 9, 30 (2013), 897–945. https://doi.org/10.4086/toc.2013.v009a030

[16] Jacek Cichoń, Mirosław Kutyłowski, and Kamil Wolny. 2017. Braid Chain Radio Communication. In *Algorithms for Sensor Systems*, Antonio Fernández Anta, Tomasz Jurdzinski, Miguel A. Mosteiro, and Yanyong Zhang (Eds.). Springer International Publishing, Cham, 223–235.

[17] Jacek Cichoń, Jakub Lemiesz, and Marcin Zawada. 2012. On Message Complexity of Extrema Propagation Techniques. In *Ad-hoc, Mobile, and Wireless Networks*, Xiang-Yang Li, Symeon Papavassiliou, and Stefan Ruehrup (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–13.

[18] Reuven Cohen, Liran Katzir, and Aviv Yehezkel. 2015. A Unified Scheme for Generalizing Cardinality Estimators to Sum Aggregation. *Inf. Process. Lett.* 115, 2 (Feb. 2015), 336–342. https://doi.org/10.1016/j.ipl.2014.10.009

[19] Graham Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *J. Algorithms* 55, 1 (April 2005), 58–75. https://doi.org/10.1016/j.jalgor.2003.12.001

[20] Anirban Dasgupta, Kevin J. Lang, Lee Rhodes, and Justin Thaler. 2016. A Framework for Estimating Stream Expression Cardinalities. In *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016 (LIPIcs)*, Wim Martens and Thomas Zeume (Eds.), Vol. 48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 6:1–6:17. https://doi.org/10.4230/LIPIcs.ICDT.2016.6

[21] Luc Devroye. 1986. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, NY, USA.

[22] Marianne Durand and Philippe Flajolet. 2003. Loglog Counting of Large Cardinalities. In *Annual European Symposium on Algorithms (ESA03) (Lecture Notes in Computer Science)*, G. Di Battista and U. Zwick (Eds.), Vol. 2832. Springer Berlin Heidelberg, 605–617.

[23] Otmar Ertl. 2017. New Cardinality Estimation Methods for HyperLogLog Sketches. *CoRR* (2017). arXiv:1706.07290 http://arxiv.org/abs/1706.07290

[24] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. In *Proceedings of the Conference on Analysis of Algorithms (AofA'07)*. 127–146.

[25] Philippe Flajolet and G. Nigel Martin. 1985. Probabilistic Counting Algorithms for Data Base Applications. *J. Comput. Syst. Sci.* 31, 2 (1985), 182–209.

[26] Frédéric Giroire. 2009. Order statistics and estimating cardinalities of massive data sets. *Discrete Applied Mathematics* 157, 2 (2009), 406–427.

[27] Stefan Heule, Marc Nunkesser, and Alex Hall. 2013. HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm. In *Proceedings of the EDBT 2013 Conference*. Association for Computing Machinery, Genoa, Italy, 683–692.

[28] Sergey Ioffe. 2010. Improved Consistent Sampling, Weighted Minhash and L1 Sketching.. In *ICDM*, Geoffrey I. Webb, Bing Liu, Chengqi Zhang, Dimitrios Gunopulos, and Xindong Wu (Eds.). IEEE Computer Society, 246–255.

[29] Donald E. Knuth. 1998. *The art of computer programming, volume 3: sorting and searching*. Addison Wesley Longman Publishing Co., Inc.

[30] Julien Le Maire, Nicolas Brunie, Florent De Dinechin, and Jean-Michel Muller. 2016. Computing floating-point logarithms with fixed-point operations. In *2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH)*. 156–163.

[31] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA.

[32] Damon Mosk-Aoyama and Devavrat Shah. 2006. Computing separable functions via gossip. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing (PODC '06)*. Association for Computing Machinery, 113–122.

[33] Anshumali Shrivastava. 2016. Exact Weighted Minwise Hashing in Constant Time. *CoRR* abs/1602.08393 (2016). arXiv:1602.08393 http://arxiv.org/abs/1602.08393

[34] Daniel Ting. 2016. Towards Optimal Cardinality Estimation of Unions and Intersections with Sketches. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. ACM, New York, NY, USA, 1195–1204. https://doi.org/10.1145/2939672.2939772

[35] Yu-Chee Tseng, Fang-Jing Wu, and Wan-Ting Lai. 2013. Opportunistic data collection for disconnected wireless sensor networks by mobile mules. *Ad Hoc Networks* 11, 3 (2013), 1150–1164. https://doi.org/10.1016/j.adhoc.2013.01.001

[36] Hang Wan, Michael David, and William Derigent. 2019. Defining the Communication Architecture for Data Aggregation in Wireless Sensor Networks: Application to Communicating Concrete Design. In *7th International Conference on Future Internet of Things and Cloud, FiCloud 2019, Istanbul, Turkey, August 26-28, 2019*, Muhammad Younas, Irfan Awan, and Takahiro Hara (Eds.). IEEE, 102–108. https://doi.org/10.1109/FiCloud.2019.00022

[37] Viktor Witkovský. 2001. Computing the Distribution of a Linear Combination of Inverted Gamma Variables. *Mathematics Preprint Archive* Vol. 2001 (June 2001), 1360–1371. Issue 6. https://ssrn.com/abstract=3162731