All-Distances Sketches, Revisited: HIP Estimators for Massive Graphs Analysis

Edith Cohen
Microsoft Research SVC
editco@microsoft.com

ABSTRACT

[1]Graph datasets with billions of edges, such as social and Web graphs, are prevalent. To be feasible, computation on such large graphs should scale linearly with graph size. All-distances sketches (ADSs) are emerging as a powerful tool for scalable computation of some basic properties of individual nodes or the whole graph.

ADSs were first proposed two decades ago (Cohen 1994) and more recent algorithms include ANF (Palmer, Gibbons, and Faloutsos 2002) and hyperANF (Boldi, Rosa, and Vigna 2011). A sketch of logarithmic size is computed for each node in the graph and the computation in total requires only a near linear number of edge relaxations. From the ADS of a node, we can estimate its neighborhood cardinalities (the number of nodes within some query distance) and closeness centrality. More generally we can estimate the distance distribution, effective diameter, similarities, and other parameters of the full graph. We make several contributions which facilitate a more effective use of ADSs for scalable analysis of massive graphs.

We provide, for the first time, a unified exposition of ADS algorithms and applications. We present the Historic Inverse Probability (HIP) estimators which are applied to the ADS of a node to estimate a large natural class of queries including neighborhood cardinalities and closeness centralities. We show that our HIP estimators have at most half the variance of previous neighborhood cardinality estimators and that this is essentially optimal. Moreover, HIP obtains a polynomial improvement for more general queries and the estimators are simple, flexible, unbiased, and elegant.

We apply HIP for approximate distinct counting on streams by comparing HIP and the original estimators applied to the HyperLogLog Min-Hash sketches (Flajolet et al. 2007). We demonstrate significant improvement in estimation quality for this state-of-the-art practical algorithm and also illustrate the ease of applying HIP.

Finally, we study the quality of ADS estimation of distance ranges, generalizing the near-linear time factor-2 approximation of the diameter.

1. INTRODUCTION

Massive graph datasets are prevalent and include social and Web graphs. Due to sheer size, computation over these graphs should scale nearly linearly with the number of edges. One task that received considerable attention is computing the distance distribution. The distance distribution of a node i contains, for each distance d, the number of nodes that are of distance d from i, that is, the cardinality of the d-neighborhood of i. The distance distribution of a graph is the number of pairs of nodes for each distance d. The distance distribution captures important properties of nodes and of the whole network, reflecting on performance and information propagation, and incorporates parameters such as node centrality, spid, and effective diameter [25, 41, 21, 8, 9, 4].

The distance distributions for all nodes can be computed through an all-pairs shortest paths, which is computationally expensive, even with state-of-the-art methods [36, 2], and not feasible for very large networks. Efficient algorithms which approximate the distance distributions were proposed in the last two decades [14, 41, 18, 21, 8]. Implementations [4] based on ANF [41] and hyperANF [8] , and more recently, [15], based on [14, 18], target social graphs with billions of edges.

At the core of all these algorithms [14, 41, 18, 21, 8] is a computation of a sketch for each node, which we call the *All Distances Sketch (ADS)*. The ADS of a node v contains a random sample of nodes, where the inclusion probability of a node u decreases with its distance from v (more precisely, inversely proportional to the number of nodes closer to v than u). For each included node, the ADS also contains its distance from v. The ADSs of different nodes are *coordinated*, which means that the inclusions of each particular node in the ADSs of other nodes are positively correlated. Coordination is an artifact of the way the ADSs are computed (we could not compute independent sketches as efficiently) but also enables further applications such as estimating similarity between neighborhoods of two nodes [14], their distance, and their closeness similarities [15].

An ADS is an extension of the simpler and better-known *Min-Hash* sketch [29, 14] (term min-wise/min-hash was coined later by Broder [13]): The ADS of a node v is essentially the union of coordinated Min-Hash sketches of all the sets of the i closest nodes to v (for all possible values of i.) Min-Hash sketches are extensively used for approximate distinct counting [29, 14, 24, 5] and similarity estimation [14, 13, 12] and come in three flavors, which correspond to sampling schemes: A k-min sketch [29, 14] is a k sample obtained with replacement, a bottom-k sketch [14] is a k sample without replacement, and a k-partition sketch [29] samples one from each of the k buckets in a random partition. All three flavors were studied because they provide different tradeoffs between update costs, information, and maintenance costs. In all three, the integer parameter $k \geq 1$ controls a tradeoff between the information content (and accuracy of attainable estimates) of the sketch and resource usage (for storing, updating, or querying the sketch). Coordination of the sketches corresponds to coordination of the underlying samples, a concept that can be traced back four decades [11]. Accordingly, ADSs come in the same three flavors: k-mins [14, 41], bottom-k [14, 18], and k-partition [8], and have expected size $\leq k \ln n$. Our detailed presentation of all flavors, provided in Section 2, facilitates a unified study of estimators.

Algorithms (see Appendix E) which compute the set of ADSs are based on classic shortest-paths algorithms: PRUNED DIJKSTRA'S [14], which performs pruned applications of Dijkstra's algorithm (or Breadth First Searches for unweighted graphs). [14], DP [41, 8], which uses dynamic programming and applies when edges are unweighted (ADS computation is implicit in [41, 21, 8], as entries are computed but not retained.), and LOCAL UPDATES, which applies dynamic programming to weighted graphs. With unweighted graphs, these algorithms perform $O(km \log n)$ edge relaxations

[1]This is a full version of a PODS 2014 paper

(where m is the number of edges and n the number of nodes), and this is also their main-memory single-processor running times. LOCAL UPDATES is node-centric and appropriate for MapReduce and similar platforms [36, 38].

Our main technical contributions are in the *estimation* component, where we use the ADSs to estimate the distance distribution, closeness centralities, and more general queries that are useful for analysis of social and other massive graphs. Our estimators, while clean and elegant, are geared for practice: getting the most from the information present in the sketch (in an exact, rather than an asymptotic sense), in terms of minimizing variance. Specifically, we use the Coefficient of Variation (CV), which is the ratio of the standard deviation to the mean,

Prior to our work, ADS-based neighborhood cardinality estimators [14, 18, 41, 28, 8] were applied by obtaining from the ADS a corresponding Min-Hash sketch of the neighborhood and applying a cardinality estimator [14, 29, 24, 28] to that Min-Hash sketch. We refer to these estimators as *basic*.

Cardinality estimators through the lens of estimation theory: In Section 3 we review Min-Hash cardinality estimators. Our exposition provides new insights from estimation theory on the optimality of these seemingly ad-hoc estimators. Our analysis also facilitates the comparison of basic estimators with the new estimators we propose here. The first-order term (and an upper bound) on the CV of the basic estimators is $1/\sqrt{k-2}$. We show, using the Lehmann-Scheffé theorem, that these estimators are the (unique) optimal unbiased estimators, in terms of minimizing variance.

Historic Inverse Probability (HIP) estimators: In Section 4 we present the novel HIP estimators, which improve over the basic estimators. The improvement is possible by utilizing all information present in the ADS (or accumulated during its computation), rather than only looking at the Min-Hash sketch of the estimated neighborhood. We show that for neighborhood cardinalities, the HIP estimators obtain a factor-2 reduction in variance over basic estimators, with CV upper bounded by the first-order term $1/\sqrt{2(k-1)}$). We further show that our HIP estimators are essentially optimal for ADS-based neighborhood cardinality estimates, and nearly match an asymptotic (for large enough cardinality) lower bound of $1/\sqrt{2k}$ on the CV. Moreover, the HIP estimates can be integrated in existing implementations (ANF [41] and hyperANF [8]) and replace the basic estimators essentially without changing the computation. We perform simulations that demonstrate a factor $\sqrt{2}$ gain in both mean square error and mean relative error of HIP over basic estimators.

Moreover, our HIP estimates have a linear form which makes them useful for an expressive general class of queries. Each node j has a nonnegative estimate $a_{ij} \geq 0$, which we refer to as *adjusted weight* on its presence with respect to i. The adjusted weight is unbiased (has expectation 1 for any j reachable from i). It is strictly positive $a_{ij} > 0$ if and only if $j \in \mathrm{ADS}(i)$, in which case it can be computed from $\mathrm{ADS}(i)$.

The cardinality of the d-neighborhood of i can be estimated by the sum of the adjusted weights of nodes in $\mathrm{ADS}(i)$ that are of distance at most d from i. More generally, we can obtain unbiased and nonnegative estimates for arbitrary queries of form

$$Q_g(i) = \sum_{j|d_{ij}<\infty} g(d_{ij}, j) \;, \tag{1}$$

where $g(j, d_{ij}) \geq 0$ is a function over both node IDs and distances. The respective estimate $\hat{Q}_g(i) = \sum_{j\in\mathrm{ADS}(i)} a_{ij} g(j, d_{ij})$ is a sum over (the logarithmically many) nodes in $\mathrm{ADS}(i)$. Choosing $g(d_{ij}) = d_{ij}$, $Q_g(i)$ is the sum of distances from i, which is (the inverse of) the classic Bavelas closeness centrality measure [6]. Decay of relevance with distances [20, 44] and meta-data based node filters are captured by queries of the form:

$$C_{\alpha,\beta}(i) = \sum_{j|d_{ij}<\infty} \alpha(d_{ij})\beta(j) \;, \tag{2}$$

where $\alpha \geq 0$ is monotone non-increasing and $\beta \geq 0$ is a nonnegative function over node IDs. The function β facilitates measuring centrality with respect to a filter applied to the meta-data of each node. For example, β can be a predicate that depends on gender, locality, or age in a social network or the topic in a Web graph. When using $\beta \equiv 1$, neighborhood cardinality is expressed using $\alpha(x) = 1$ if $x \leq d$ and $\alpha(x) = 0$ otherwise. Choosing $\alpha(x) \equiv 1$ gives the number of reachable nodes from i, $\alpha(x) = 2^{-x}$ gives exponential attenuation with distance [22], and $\alpha(x) = 1/x$ gives the (inverse) harmonic mean of distances from i [40, 10]).

In [20, 17] we estimated (2) from the ADS of i for any (non-increasing) α. The handling of a general β, however, required an ADS computation specific to β (see Appendix B). We obtained unbiased nonnegative estimators through a reduction to basic neighborhood cardinality estimators, with the same CV of $1/\sqrt{k-2}$. On the same problem, our ADS HIP estimators:

$$\hat{C}_{\alpha,\beta}(i) = \sum_{j\in\mathrm{ADS}(i)} a_{ij}\alpha(d_{ij})\beta(j) \;, \tag{3}$$

have CV upper bounded by $1/\sqrt{2(k-1)}$. Moreover, we are also able to obtain unbiased estimates for general queries when the filter β in (2) (or the function g in (1)) are specified after the sketches are computed. This flexibility of using the same set of sketches for many queries is important in many conceivable applications of social networks or Web graphs analysis. For such queries, our HIP estimators obtain up to an (n/k)-fold improvement in variance over state of the art, which we believe is a subset-weight estimator applied to the Min-Hash sketch of all reachable nodes (by taking the average of $g(d_{ij}, j)$ over the k samples, multiplied by a cardinality estimate of the number of reachable nodes n).

HIP estimators for approximate distinct counting: Almost all streaming distinct counters [29, 14, 24, 5, 32, 30] maintain a Min-Hash sketch of the distinct elements. To answer a query (number of distinct elements seen so far), a "basic" estimator is applied to the sketch. In Section 5 we instead apply our HIP estimators. To do that, we consider the sequence of elements which invoked an update of the Min-Hash sketch over time (this corresponds to entries in the ADS computed with respect to distance rather than time). Even though the entry is not retained, (the streaming algorithm only retains the Min-Hash sketch), we can compute the adjusted weight of the new distinct element that invoked the update. These adjusted weights are added up to obtain a running estimate. To apply HIP, we therefore need to maintain the Min-Hash sketch and an additional approximate (non-distinct) counter, which maintains an approximate count of distinct elements. The approximate counter is updated (by a positive amount which corresponds to the adjusted weight of the element) each time the sketch is updated.

We experimentally compare our HIP estimator to the HyperLogLog approximate distinct counter [28], which is considered to be the state of the art practical solution. To facilitate comparison, we apply HIP to the same Min-Hash sketch with the same parametrization that the HyperLogLog estimator was designed for. Nonetheless, we demonstrate significantly more accurate estimates using HIP. Moreover, our HIP estimators are unbiased, principled, and do not require ad-hoc corrections. They are flexible in that they apply to all Min-Hash sketch flavors and can be further parametrized

according to application needs or to obtain even better accuracy for the same memory.

Permutation estimators: The basic and HIP estimators have CV that is essentially independent of cardinality (the neighborhood size). When we have an upper bound on the domain size (total number of nodes), we can improve our cardinality estimates for sets that comprise a good fraction of the domain. The *permutation estimator* (presented in Section 4.4) is a variation on our HIP estimators. We experimentally show that permutation improves over plain HIP when the cardinality is a good fraction (at least 20%) of the total number of nodes.

Estimation quality for distance ranges: The *cummulative* distance distribution, that is, the number of pairs within distance at most d, can be estimated with small relative error. For the number of pairs of distance *equal to* d, we do not have the same guarantees. The estimators are nonnegative and unbiased, and in practice have small relative error [21, 8], but since the problem generalizes estimating the graph diameter [1, 42] we can not expect theoretical guarantees using our near-linear time sketch computation.

In Section 6 we explore the estimation quality for exact distances. For directed and undirected graphs, we explain the good performance in practice by the expansion of "real" graphs. For undirected graphs, we provide a guarantee that holds regardless of expansion: We show that for any two nodes (i, j), our estimate on the number of pairs with an endpoint in $\{i, j\}$ with distance in $[d_{ij}/2, 3d_{ij}/2]$ has CV that is $O(1/\sqrt{k})$. This result extends the best-known near-linear time diameter approximation factor of 2.

2. ALL-DISTANCES SKETCHES

We start with a brief review of Min-Hash sketches. The Min-Hash sketch summarizes a subset N of items (from some domain U) and comes in three flavors, k-mins, k-partition, and bottom-k, where the parameter k determines the sketch size.

The sketch is randomized and defined with respect to (one or more, depending on flavor) random permutations of the domain U. It is convenient to specify a permutation by assigning random *rank* values, $r(j) \sim U[0, 1]$, to items. The permutation is the list of items sorted by increasing rank order. To specify multiple permutations, we use multiple rank assignments. A k-mins sketch [29, 14] includes the item of smallest rank in each of k independent permutations and corresponds to sampling k times with replacement. A k-partition sketch [29, 28, 35] first maps items uniformly at random to k buckets and then includes the item with smallest rank in each bucket. A bottom-k sketch [14, 12] (also known as KMV sketch [5], coordinated order samples [11, 43, 39], or CRC [34]) includes the k items with smallest rank in a single permutation and corresponds to sampling k times without replacement. For $k = 1$, all three flavors are the same.

Min-Hash sketches of different subsets N are *coordinated* if they are generated using the same random permutations (or mappings) of the domain U. The notion of coordination can be traced to [11] and in the CS literature to [14, 12].

Before continuing to graphs, we introduce some terminology. For a set N and a numeric function $r : N$, the function $\mathrm{k}^{\mathrm{th}}_r(N)$ returns the k^{th} smallest value in the range of r on N. If $|N| < k$ then we define $\mathrm{k}^{\mathrm{th}}_r(N)$ to be the supremum of the range of r (we mostly use $r \in [0, 1]$ and the supremum is 1.) We consider directed or undirected, weighted or unweighted graphs. For nodes i, j, let d_{ij} be the distance from i to j. For an interval J and node i, $N_J(i) = \{j | d_{ij} \in J\}$ is the set of nodes with distance in J from i, and $n_J(i) = |N_J(i)|$ is the cardinality of $N_J(i)$. For a distance d, we use the shorthand $N_{[0,d]} \equiv N_d$ and $n_{[0,d]} \equiv n_d$. We use the notation $\Phi_{<j}(i)$ for the set of nodes that are closer to node i than node j and $\pi_{ij} = 1 + |\Phi_{<j}(i)|$ for the *Dijkstra rank* of j with respect to i (j's position in the nearest neighbors list of i).

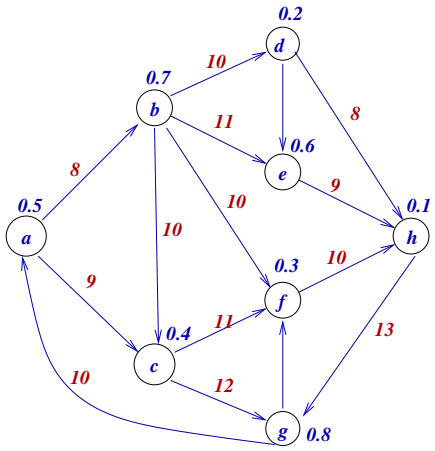

Figure 1: A directed graph with random rank values associated with its nodes.

The ADS of a node i, $\mathrm{ADS}(i)$, is a set of node ID and distance pairs. The included nodes are a sample of the nodes reachable from i and with each included node $j \in \mathrm{ADS}(i)$ we store the distance d_{ij}. $\mathrm{ADS}(i)$ is the union of coordinated Min-Hash sketches of the neighborhoods $N_d(i)$ (for all possible values of d). The ADSs are defined with respect to random mappings/permutations of the set of all nodes and come in the same three flavors, according to the underlying Min-Hash sketches: Bottom-k, k-mins, and k-partition. For $k = 1$, all flavors are equivalent. [2] For simplicity, our definitions of $\mathrm{ADS}(i)$ assume that distances d_{ij} are unique for different j (Which can be achieved using tie breaking). A definition which does not use tie breaking is given in Appendix D.

A *bottom-k* ADS [18] is defined with respect to a single random permutation. $\mathrm{ADS}(i)$ includes a node j if and only if the rank of j is one of the k smallest ranks amongst nodes that are at least as close to i:

$$j \in \mathrm{ADS}(i) \iff r(j) < \mathrm{k}^{\mathrm{th}}_r(\Phi_{<j}(i)) \,. \tag{4}$$

A *k-partition* ADS (implicit in [8]) is defined with respect to a random partition $\textsc{bucket} : V \to [k]$ of the nodes to k subsets $V_h = \{i | \textsc{bucket}(i) = h\}$ and a random permutation. The ADS of i contains j if and only if j has the smallest rank among nodes in its bucket that are at least as close to i.

$$j \in \mathrm{ADS}(i) \iff r(j) < \min\{r(h) \mid \textsc{bucket}(h) = \textsc{bucket}(j) \wedge h \in \Phi_{<j}(i)\} \,.$$

A k-mins ADS [14, 41] is simply k independent bottom-1 ADSs, defined with respect to k independent permutations.

As mentioned, the term All-Distances Sketch reflects the property that the sketch "contains" a Min-Hash sketch with respect to any distance d. We briefly explain how a Min-Hash sketch of a neighborhood $N_d(v)$ can be obtained from the ADS of a node v. This can be done for any $d \geq 0$ and in the context of Min-Hash sketches, $N_d(v)$ is treated as a subset of nodes.

For a k-mins ADS, we are interested in the k-mins Min-Hash sketch of $N_d(v)$ which is, for each of the k permutations r, $x \leftarrow$

[2] The term *least element lists* was used for ADS in [14].

$\min_{u \in N_d(v)} r(u)$. The value for a given permutation is the minimum rank of a node of distance at most d in the respective bottom-1 ADS. The k minimum rank values $x^{(t)}$ $t \in [k]$ we obtain from the different permutations are the k-mins Min-Hash sketch of $N_d(v)$. Similarly, the bottom-k Min-Hash sketch of $N_d(v)$ includes the k nodes of minimum rank in $N_d(v)$, which are also the k nodes of minimum rank in $\mathrm{ADS}(v)$ with distance at most d. A k-partition Min-Hash sketch of $N_d(v)$ is similarly obtained from a k-partition ADS by taking, for each bucket $i \in [k]$, the smallest rank value in $N_d(v)$ of a node in bucket i. This is also the smallest value in $\mathrm{ADS}(v)$ over nodes in bucket i that have distance at most d from v.

Some of our analysis assumes that the rank $r(j)$ and (for k-partition ADSs) the bucket $\textsc{bucket}(j)$ are readily available for each node j. This can be achieved using random hash functions.

For directed graphs, we consider both the *forward* and the *backward* ADS, which are specified with respect to forward or reverse paths from i. When needed for clarity, we augment the notation with $\overrightarrow{X}$ (forward) and $\overleftarrow{X}$ (backward) when X is the ADS, N, or n.

EXAMPLE 2.1. *Consider the graph of Figure 1. To determine the forward ADS of node a, we sort nodes in order of increasing distance from a. The order is a, b, c, d, e, f, g, h with respective distances $(0, 8, 9, 18, 19, 20, 21, 26)$. For $k = 1$, the (forward) ADS of a is: $\overrightarrow{ADS}(a) = \{(0, a), (9, c), (18, d), (26, h)\}$. The first value in each pair is the distance from a and the second is the node ID. To compute the reverse ADS of b, we look at nodes in sorted reverse distance from b: b, a, g, c, h, d, e, f with respective reverse distances $(0, 8, 18, 30, 31, 39, 40, 41)$. We obtain $\overleftarrow{ADS}(b) = \{(0, b), (8, a), (30, c), (31, h)\}$. The bottom-2 forward ADS of a contains all nodes that have one of the 2 smallest ranks in the prefix of the sorted order: so it also includes $\{(8, b), (20, f)\}$.*

The expected number of nodes in $\mathrm{ADS}(i)$ is $\leq k \ln(n)$, where n is the number of reachable nodes from i: This was established in [14] for k-mins ADS and in [19] for bottom-k ADS.

LEMMA 2.2. *[14, 19] The expected size of a bottom-k ADS is*

$$k + k(H_n - H_k) \approx k(1 + \ln n - \ln k) \ ,$$

where $H_i = \sum_{j=1}^{i} 1/j$ is the ith Harmonic number and n is the number of nodes reachable from v. The expected size of a k-partition ADS is accordingly $kH_{n/k} \approx k(\ln n - \ln k)$.

PROOF. For bottom-k ADS, we consider the nodes sorted by increasing distance from v, assuming unique distances. The ith node is included in the bottom-k ADS of v with probability $p_i = \min\{1, k/i\}$. Node inclusions are independent (when distances are unique, but otherwise are negatively correlated). The expected size of the ADS of v is the sum of node inclusions which is

$$\sum_{i=1}^{n} p_i = k + k(H_n - H_k) \ .$$

Similarly, for k-partition, (assuming a random partition and permutation), the expected number of included nodes from each bucket is $\ln(n/k)$ (since each bucket includes in expectation n/k nodes) and therefore the total expected size is $k \ln(n/k)$. □

Base-b ranks: The ADS definition as provided includes node IDs, that is, unique identifiers for nodes. Unique IDs are of size $\lceil \log_2 n \rceil$ and allow us to obtain unique ranks and also support queries involving meta-data based node selections. For many queries, including neighborhood cardinality estimation, we can use ranks that have a smaller representation: For some base $b > 1$, we use the rounded rank values $r'(j) = b^{-h_j}$, where $h_j = \lceil -\log_b r(j) \rceil$. The rounded rank can be represented by the integer h_j. The value of the base b trades-off the sketch size and the information it carries, where both increase when b is closer to 1.

With base-b ranks, the expected value of the largest h_j, which corresponds to the smallest $r(j)$, is $\log_b n$. Thus, the representation size of the rounded smallest rank is $\log_2 \log_b n$. The expected deviation from the expectation is $\leq \log_b 2$, which means that a set of k smallest ranks in a neighborhood or the k smallest ranks in different permutations can be compactly represented using an expected number of $\log_2 \log_2 n + k \log_b 2$ bits.

In the sequel, we consider full ranks and then point out the implication of using base-b ranks.

3. MIN-HASH CARDINALITY ESTIMATE

In this section we review estimators for the cardinality $|N| = n$ of a subset N that are applied to a Min-Hash sketch of N.

The cardinality of $N_d(v)$ can be estimated by obtaining its Min-Hash sketch from $\mathrm{ADS}(v)$ and applying a cardinality estimator to this Min-Hash sketch. This also applies to directed graphs, in which case we can estimate the size of the outbound d-neighborhood $\overrightarrow{n}_d(v)$ from $\overrightarrow{\mathrm{ADS}}(v)$ and similarly estimate the size of the inbound d-neighborhood $\overleftarrow{n}_d(v)$ from $\overleftarrow{\mathrm{ADS}}(v)$.

As mentioned in the introduction, we use the CV to measure the quality of the estimates. The CV of an estimator $\hat{n}$ of n is $\sqrt{\mathsf{E}[(n - \hat{n})^2]}/n$. For the same value of the parameter k, the bottom-k sketch contains the most information, but all flavors are similar when $n \gg k$. We first consider full precision ranks and then explain the implication of working with base-b ranks. For illustrative purposes, we start with the k-mins sketch. We then consider the more informative bottom-k sketch. The lower bound for the k-partition sketch is implied by the bound for the other flavors.

3.1 k-mins estimator

The k-mins sketch has the vector form x_i $i \in [k]$. The cardinality estimator $\frac{k-1}{\sum_{i=1}^{k} -\ln(1-x_i)}$ was presented and analysed in [14]. It is unbiased for $k > 1$. Its variance is bounded only for $k > 2$ and the CV is equal to $1/\sqrt{k-2}$. The Mean Relative Error (MRE) is

$$\frac{2(k-1)^{k-2}}{(k-2)!\exp(k-1)} \approx \sqrt{\frac{2}{\pi(k-2)}} \ .$$

This estimator can be better understood when we view the ranks as exponentially distributed with parameter 1 (rather than uniform from $U[0, 1]$). This is equivalent, as we can use a simple 1-1 monotone transformation $y = -\ln(1 - x)$ which also preserves the Min-Hash definition. In this light, the minimum rank is exponentially distributed with parameter n. Our estimation problem is to estimate the parameter of an exponential distribution from k independent samples and we use the estimator $\frac{k-1}{\sum_{i=1}^{k} y_i}$, where $y_i = -\ln(1 - x_i)$.

We now apply classic estimation theory to better understand how well this estimator uses the information available in the Min-Hash sketch.

LEMMA 3.1. *Any unbiased estimator applied to the k-mins Min-Hash sketch must have CV that is at least $1/\sqrt{k}$.*

PROOF. For cardinality n, each of the k entries (minimum ranks) is an exponentially distributed random variable and therefore has density function ne^{-nx}.

Since entries in the k-mins sketch are independent, the density function (likelihood function) of the sketch is the product of the k density functions $f(\boldsymbol{y}; n) = n^k e^{-n\sum_{i=1}^k y_i}$. Its logarithm, the log likelihood function, is $\ell(\boldsymbol{y}; n) = k \ln n - n\sum_{i=1}^k y_i$. The Fisher information, $I(n)$, is the negated expectation of the second partial derivative of $\ell(\boldsymbol{y}; n)$ (with respect to the estimated parameter n). We have

$$\frac{\partial^2 \ell(\boldsymbol{y}; n)}{\partial^2 n} = -\frac{k}{n^2} .$$

This is constant, and equal to its expectation. Therefore $I(n) = k/n^2$.

We now apply the Cramér-Rao lower bound which states that the variance of any unbiased estimator is at least the inverse of the Fisher information: $\frac{1}{I(n)} = \frac{n^2}{k}$. A corresponding lower bound of $\frac{1}{\sqrt{k}}$ on the CV is obtained by taking the square root and dividing by n. □

We next show that the sum $\sum_{i=1}^k y_i$ captures all necessary information to obtain a minimum variance estimator for n.

LEMMA 3.2. *The sum of the minimum ranks $\sum_{i=1}^k y_i$ is a sufficient statistics for estimating n from a k-mins sketch.*

PROOF. The likelihood function $f(\boldsymbol{y}; n) = n^k e^{-n\sum_{i=1}^k y_i}$ depends on the sketch only through the sum $\sum_{i=1}^k y_i$. □

Therefore, from the Rao-Blackwell Theorem [7], a minimum variance estimator applied to the sketch may only depend on $\sum_{i=1}^k y_i$. We can further show that $\sum_{i=1}^k y_i$ is in fact a *complete* sufficient statistics. A sufficient statistics T is complete if any function g for which $E[g(T)] = 0$ for all n must be 0 almost everywhere (with probability 1). The Lehmann-Scheffé Theorem [33] states that any unbiased estimator which is a function of a complete sufficient statistics must be the unique Uniform Minimum Variance Unbiased Estimator (UMVUE). Since our estimator is unbiased, it follows that it is the unique UMVUE. That is, there is no other estimator which is unbiased and has a lower variance! (for any value of the parameter n).

This optimality results provides an interesting insight to the thread of research on approximate distinct counting (and to practice). One can easily come up with several ways of using the sketch information to obtain an estimator: taking the median, averaging quantiles, removing the two extreme values, and so on. The median specifically had been considered [14, 3, 5] because it is more amenable to obtaining concentration bounds. We now understand that while these estimators can have variance that is within a constant factor of optimal, estimation theory shows that (in terms of variance) the average, and the average alone, carries all the information we need and anything else is strictly inferior.

3.2 Bottom-k estimator

The bottom-k estimator includes the k smallest rank values in N, and we use the estimator $\frac{k-1}{\tau_k}$, where $\tau_k = \mathrm{k}_r^{\mathrm{th}}(N)$ is the kth smallest rank in N. This estimator is a conditional inverse-probability estimator [31]: For each element in N we consider its probability of being included in the Min-Hash sketch, conditioned on fixed ranks of all other elements. This estimator is therefore unbiased. The conditioning was applied with priority sampling [23] (bottom-k [19]) subset sum estimation.

The information content of the bottom-k sketch is strictly higher than the k-mins sketch [18]. We show that the CV of this estimator is upper bounded by the CV of the k-mins estimator:

LEMMA 3.3. *The bottom-k estimator has CV $\leq 1/\sqrt{k-2}$.*

PROOF. We interpret the bottom-k cardinality estimator as a sum of n negatively correlated inverse-probability [31] estimates for each element, which estimate the presence of the element in N. (That is, for each $v \in N$, estimating its contribution of "1" to the cardinality and for each $v \notin N$, estimating 0). The inclusion probability of an element is with respect to fixed ranks of all other elements. In this case, an element is one of the $k-1$ smallest ranks only if its rank value is strictly smaller than the $k-1$ smallest rank amongst the $n-1$. For elements currently in the sketch, this threshold value is τ_k. These estimates (adjusted weights) are equal and positive only for the $k-1$ elements of smallest rank. The variance of the adjusted weight conditioned on fixing the rank values of other elements is $1/p - 1$, where p is the probability that the rank of our element is smaller than the threshold.

When ranks are exponentially distributed (which is convenient choice for analysis), the distribution of the $k-1$ smallest amongst $n-1$ is the sum of $k-1$ exponential random variables with parameters $n-1, n-1, \ldots, n-k+1$. We denote the density and CDF functions of this distribution by $b_{n-1,k-1}$ and $B_{n-1,k-1}$, respectively. We have $p = 1 - \exp(-x)$ and the adjusted weight of each element has variance of $1/p - 1 = \frac{\exp(-x)}{1-\exp(-x)}$ (conditioned on x). We now compute the expectation of the variance according to the distribution of x.

We denote by $s_{n,k}$ and $S_{n,k}$ the respective distribution function of the sum of k exponentially distributed random variables with parameter n.

$$\begin{aligned}
\mathsf{Var}[\hat{a}_v] &= \int_0^\infty b_{n-1,k-1}(x)\frac{e^{-x}}{1-e^{-x}}dx \leq \int_0^\infty s_{n-1,k-1}(x)\frac{1}{x}dx \\
&= \int_0^\infty \frac{(n-1)^{k-1}x^{k-2}}{(k-2)!}e^{-(n-1)x}\frac{1}{x}dx \\
&= \frac{(n-1)^{k-1}}{(k-2)!}\frac{(k-3)!}{(n-1)^{k-2}} = \frac{n-1}{k-2} .
\end{aligned}$$

The first inequality follows from $\frac{e^{-x}}{1-e^{-x}} \leq 1/x$ and $\forall x, B_{n,k}(x) \leq S_{n,k}(x)$, that is, $B_{n,k}$ is dominated by the sum of k exponential random variables with parameter n. We then substitute the probability density function [26] (also used for analyzing the k-mins estimator in [14])

$$s_{n,k} = \frac{n^k x^{k-1}}{(k-1)!}e^{-nx} .$$

The second to last equality uses $\int_0^\infty x^a e^{-bx}dx = a!/b^{a+1}$ for natural a, b,

Estimates for different elements are negatively correlated (an element being sampled makes is slightly less likely for another to be sampled) and thus, the variance on the cardinality estimate is at most the sum of variances of the $n-(k-1)$ elements with positive variance, (the first $k-1$ have 0 variance), which is $(n-k+1)(n-1)/(k-2)$, obtaining that the CV is at most

$$\sqrt{\frac{1-\frac{kn-k+1}{n^2}}{k-2}} \leq \sqrt{\frac{1}{k-2}} .$$

□

The improvement of bottom-k over the k-mins estimator is more pronounced when the cardinality n is smaller and closer to k. The first order term, however, is the same and when $n \gg k$, the CV of the bottom-k estimator approaches $\sqrt{\frac{1}{k-2}}$.

We now consider this estimator from the estimation theoretic lens. When $n \leq k$, the variance is clearly 0. Therefore, any meaningful lower bound must depend on both n, k.

LEMMA 3.4. *Any unbiased estimator applied to the bottom-k Min-Hash sketch must satisfy*

$$\lim_{n\to\infty} CV(n,k) \geq 1/\sqrt{k}\,.$$

PROOF. Let $x_1, x_2, \ldots, x_k$ be the k smallest ranks in increasing order. From basic properties of the exponential distribution, the minimum rank $y_0 \equiv x_1$ is exponentially distributed with parameter n. For $i > 0$, the difference between the ith smallest and the $i-1$th smallest ranks, $y_i \equiv x_{i+1} - x_i$, is exponentially distributed with parameter $n - i$. Moreover, these differences y_i are independent. We can therefore equivalently represent the information in the bottom-k sketch by $(y_0, \ldots, y_{k-1})$, where y_i is independently drawn from an exponential distribution with parameter $n - i$. The joint density function is the product $f(\boldsymbol{y}; n) = \prod_{i=0}^{k-1}(n-i)e^{-(n-i)y_i}$. The Fisher information is $I(n) = \sum_{i=0}^{k} \frac{1}{(n-i)^2}$. We obtain a lower bound on the CV of at least $\frac{1}{\sqrt{\sum_{i=0}^{k-1} \frac{n^2}{(n-i)^2}}}$. When $n \gg k$, the expression approaches $\frac{1}{\sqrt{k}}$. □

LEMMA 3.5. *x_k (the kth smallest rank) is a sufficient statistics for estimating n from a bottom-k sketch.*

PROOF. We can express the joint density function $f(\boldsymbol{y}; n)$ as a product of an expression that does not depend on the estimated parameter n and $e^{-n\sum_{i=0}^{k-1} y_i} \prod_{i=0}^{k-1}(n-i)$. Therefore, $x_k = \sum_{i=0}^{k-1} y_i$ is a sufficient statistics. □

From Rao-Blackwell Theorem, the kth minimum rank captures all the useful information contained in the bottom-k sketch for obtaining a minimum variance estimator for n. Since it is a complete sufficient statistics, and our estimator is unbiased, it follows from the Lehmann-Scheffé theorem [33] that it is the unique UMVUE.

3.3 k-partition estimator

The estimator examines the $1 < k' \leq k$ nonempty buckets, and is conditioned on k'. The size of each bucket has distribution $1 + B(n - k', 1/k')$, where B is the Binomial distribution. We can approximate bucket sizes by n/k' and apply the k'-mins estimator (analysis holds for k' equal buckets). The estimate is $\frac{k'(k'-1)}{-\sum_{t=1}^{k} \ln(1-x^{(t)})}$, where $x^{(t)}$ is the minimum rank in partition t.

When $n \gg k$, the k-partition estimator performs similarly to the bottom-k and k-mins estimators. When $n < k$, there are effectively only $k' < k$ nonempty buckets. Even when $n = O(k)$, the expected size of k' is significantly smaller than k, and the CV is more similar to that of a k'-mins estimate, and therefore, can be expected to be $\sqrt{k/k'}$ larger. Moreover, the k-partition estimator is biased down: In particular, when $k' = 1$, an event with positive probability, the estimate is 0. The probability of $k' = 1$ for cardinality n is $p = (1/k)^{n-1}$. Since we do not generally know n, we can not completely correct for this bias.

3.4 Min-Hash sketches with base-b ranks

We considered cardinality estimators for sketches with "full" ranks taken from the domain $[0, 1]$. If we work with truncated ranks but ensure that there are no rank collisions, the full-rank estimators can be applied by uniformly at random "filling in" missing bits to the desired precision or better yet, computing the expectation of the estimator over these random completions. A hash range of size n^c and representation $c \log n$ implies that with probability $1/n^{c-1}$ there are no rank collisions between any two nodes in a set of size n.

A uniform random completion of the truncated ranks is an equivalent replacement to the full rank when all elements with the same base-b rank that "contend" for a sketch entry are actually represented in the sketch. If this condition does not hold, the expected full-rank completions are more likely to be smaller than uniform completions and estimates obtained with uniform completion will be biased.

To satisfy this condition we need to ensure that there are no rank collisions along the "inclusion" threshold. With bottom-k this means that the base-b kth smallest rank is strictly smaller than the base-b $(k+1)$th smallest rank. With k-mins (k-partition) it means that the base-b minimum is unique in each permutation (bucket).

If we choose $b = 1 + 1/k^c$, probability of collision is at most $1/k^{c-1}$. In this case, the expected size of the (integer exponent of the) minimum rank is $\log_2 \log_b n \approx \log_2 \log_2 n + \log_2 k^c \approx \log_2 \log_2 n + c \log k$. Moreover, we recall that the expected size of the offset from this expectation is constant times $\log_b 2$. Substituting $b \approx 1/k^c$ we obtain an expected offset of the order of $c \log k$, so we can compactly represent the sketch using $\log_2 \log_2 n + ck \log k$ bits.

If we work with a larger base, collisions are more likely and introduce bias. The estimators then need to compensate for the bias. Specialized estimators for base-2 ranks with k-mins sketches were proposed in [29] and for k-partition sketches in [28]. The HIP estimators we present next naturally apply with base-b ranks.

4. THE HIP ESTIMATOR

The *Historic Inverse Probability (HIP)* estimators we present here gain from using the complete information present in $\mathrm{ADS}(v)$ rather than extracting from it a Min-Hash sketch of the neighborhood whose size we want to estimate, and apply a cardinality estimator to that sketch. HIP estimators can be computed for all three ADS flavors and naturally extend to base-b ranks. We show that the HIP estimators obtain a factor 2 improvement in variance over the respective basic estimator and also show that they are asymptotically optimal. We also present a variant of HIP, the *permutation* cardinality estimator, which applies to bottom-k ADSs when ranks are a strict permutation of a domain $[n]$. This estimator improves over plain HIP when the cardinality is good fraction of n.

The HIP estimator is computed by scanning entries in $\mathrm{ADS}(i)$ in order of increasing distance from i. For each node $j \in \mathrm{ADS}(i)$ we compute an estimate $a_{ij} > 0$ on its presence in $\mathrm{ADS}(i)$ which we call the *adjusted weight* of j. These adjusted weights are *conditioned* inverse probability estimates, a twist on a classic Horvitz-Thompson [31] estimator which applies it conditioned on the ranks of nodes that are closer to i than j. A similar conditioning technique, in a different context, was used in [23, 19]. The adjusted weight $a_{ij} \geq 0$ has expectation $\mathsf{E}[a_{ij}] = 1$ and is positive if and only if $j \in \mathrm{ADS}(i)$.

As noted in the introduction, we can estimate $Q_g(i)$ (see (1)) from $\mathrm{ADS}(i)$ using

$$\hat{Q}_g(i) = \sum_j a_{ij} g(d_{ij}, j) = \sum_{j \in \mathrm{ADS}(i)} a_{ij} g(d_{ij}, j)\,. \qquad (5)$$

Unbiasedness follows from linearity of expectation, since each adjusted weight is unbiased. The second equality holds since only nodes $j \in \mathrm{ADS}(i)$ have positive $a_{ij} > 0$. We note that the estimate can be easily computed from $\mathrm{ADS}(i)$, since for each included node j we have the distance d_{ij}.

When we are only interested in queries Q_g where $g(d_{ij})$ only depends on the distance and not on the node ID j, we can compress

the ADS representation to a list of distance and adjusted weights pairs: For each unique distance d in $\mathrm{ADS}(i)$ we associate an adjusted weight equal to the sum of the adjusted weights of included nodes in $\mathrm{ADS}(i)$ with distance d.

To finish the presentation of the HIP estimators, we need to explain how the adjusted weights are computed for $j \in \mathrm{ADS}(i)$. We focus in detail on bottom-k ADSs and start with full ranks $r(i) \sim U[0,1]$.

4.1 HIP estimate for bottom-k ADS

Consider a node v and list nodes by increasing Dijkstra rank with respect to v, that is node i has $\pi_{vi} = i$.

For node i, we define the threshold value

$$\tau_i = \mathrm{k}^{\mathrm{th}}_r\{\Phi_{<i}(v) \cap \mathrm{ADS}(v)\} \ . \tag{6}$$

The adjusted weights a_{vi} for node i are 0 if $i \notin \mathrm{ADS}(v)$ and $1/\tau_i$ if $i \in \mathrm{ADS}(v)$. Note that τ_i, and hence a_{vi}, can be computed from $\mathrm{ADS}(v)$ for all $i \in \mathrm{ADS}(v)$.

The adjusted weights are inverse-probability estimates with respect to the probability τ_i of including i in $\mathrm{ADS}(v)$, conditioned on fixing the ranks of the nodes $1, \ldots, i-1$:

LEMMA 4.1. *Conditioned on fixed rank values of all nodes in $\Phi_{<i}(v)$, the probability of $i \in \mathrm{ADS}(v)$ is τ_i.*

PROOF. Node i is included if and only if $r(i) < \mathrm{k}^{\mathrm{th}}_r\{\Phi_{<i}(v)\}$, that is, i's rank is smaller than the kth smallest rank amongst nodes that are closer to v than i. Note that it is the same as the kth smallest rank among nodes that are in $\mathrm{ADS}(v)$ and closer to v than i, since $\mathrm{ADS}(v)$ must include all these nodes. When $r(i) \sim U[0,1]$, this happens with probability τ_i. □

Since these are inverse-probability weights, they are clearly unbiased when $\tau_i > 0$, which happens with probability 1. Note that for $i \leq k$ (when i is one of the k closest nodes to v), by definition $i \in \mathrm{ADS}(v)$, $\tau_i \equiv 1$, and therefore $a_{vi} = 1$, since the first k nodes are included with probability 1. Also note that the adjusted weights of nodes in $\mathrm{ADS}(v)$ are increasing with the distance d_{vi} (or Dijkstra rank π_{vi}). This is because the inclusion probability in the ADS decreases with distance. In particular this means that the variance of a_{vi} increases with d_{vi}.

We show that the variance of the HIP neighborhood cardinality estimator is at least a factor of 2 smaller than the variance of the basic bottom-k cardinality estimator, which in turn dominates the basic k-mins estimator.

THEOREM 4.1. *The CV of the ADS HIP estimator for a neighborhood of size n is*

$$\leq \frac{\sqrt{1 - \frac{n+k(k-1)}{n^2}}}{\sqrt{2(k-1)}} \leq \frac{1}{\sqrt{2(k-1)}} \ .$$

PROOF. When $n_d(v) \leq k$, the estimate is exact (variance is 0). Otherwise, (assuming nodes are listed by Dijkstra ranks $\pi_{vi} \equiv i$), the variance on i is $\mathsf{E}[1/p - 1]$ where p is the probability that the rank of v_i is smaller than the kth smallest rank among $v_1, \ldots, v_{i-1}$. We adapt the analysis of Lemma 3.3 for the variance of the bottom-k estimator. We use exponentially distributed ranks, and have, conditioned on kth smallest rank τ_i in $\Phi_{<i}(v)$ variance $\exp(-\tau_i)/(1-\exp(-\tau_i))$. We compute the expectation of the variance for τ_i distributed according to $b_{i-1,k}$. This is a similar computation to the proof of Lemma 3.3 and we obtain that the variance of the adjusted weight a_{vi} is bounded by $\frac{i-1}{k-1}$. Estimates for different i are again negatively correlated and thus the variance of the neighborhood estimate on n is upper bounded by $\sum_{i=k+1}^{n} \frac{i-1}{k-1} = \frac{n^2-n-k^2-k}{2(k-1)}$ and the upper bound on the CV follows. □

The bound of Theorem 4.1 extends to distance-decay closeness centralities.

COROLLARY 4.2. *For a monotone non-increasing $\alpha(x) \geq 0$ (we define $\alpha(\infty) = 0$), $\hat{C}_\alpha(i) = \sum_{j \in \mathrm{ADS}(i)} a_{ij}\alpha(d_{ij})$ is an unbiased estimator of $C_\alpha(i) = \sum_j \alpha(d_{ij})$ with CV that is at most $1/\sqrt{2(k-1)}$.*

The Corollary holds for the more general form (2) when ADSs are computed with respect to the node weights $\beta(i)$, see Appendix B. Otherwise, when estimating $Q_g(v)$ using (5), the variance is upper bounded as follows:

COROLLARY 4.3.

$$\mathsf{Var}[\hat{Q}_g(v)] \leq \sum_{i|v \rightsquigarrow i \wedge \pi_{vi} > k} g(i, d_{vi})^2 \frac{\pi_{vi} - 1}{k-1} \ .$$

In contrast, we can consider the variance of the naive estimator for $Q_g(v)$ that is mentioned in the introduction. That estimator uses a Min-Hash sketch, which is essentially a random sample of k reachable nodes. Since inclusion probabilities are about $\approx k/n$ The variance in this case is about $\frac{n-1}{k-1}\sum_i g(i, d_{vi})^2$. We can see that when $g(i, d_{vi})$ are concentrated (have higher values) on closer nodes, which the Min-Hash sketch is less likely to include, the variance of the naive estimate can be up to a factor of n/k higher, where n is the number of reachable nodes from v.

4.2 HIP estimate for k-mins and k-partition

We briefly present the HIP estimators for k-mins and k-partition ADS. Similarly, the adjusted weight is 0 if $i \notin \mathrm{ADS}(v)$ and is $1/\tau_i$ otherwise, where the inclusion threshold τ_i is computed as follows. For k-mins, a node i is included in $\mathrm{ADS}(v)$ only if it has rank value strictly smaller than the minimum rank in $\Phi_{<i}(v)$ in at least one of the k assignments r_h $h \in [k]$. Conditioned on fixed ranks of all the nodes $\Phi_{<i}(v)$, the inclusion threshold is

$$\tau_i = 1 - \prod_{h=1}^{k}(1 - \min_{j \leq i-1} r_h(j)) \ . \tag{7}$$

For k-partition ADS, we again fix both the rank values and the partition mapping (to one of the k buckets $V_1, \ldots, V_k$) of all nodes in $\Phi_{<i}(v)$. We then compute the inclusion threshold, which is the probability that $i \in \mathrm{ADS}(v)$ given that conditioning. This is with respect to a uniform mapping of node i to one of the k buckets and random rank value. We obtain inclusion threshold

$$\tau_i = \frac{1}{k}\sum_{h=1}^{k} \min_{j \in V_h \cap \Phi_{<i}(v)} r(j) \ , \tag{8}$$

defining the minimum rank over an empty set $V_h \cap \Phi_{<i}(v)$ to be 1. Note that the threshold τ_i, and therefore the respective adjusted weight a_{vi}, can be computed from $\mathrm{ADS}(v)$.

4.3 Lower bound on variance

We show that the variance of the HIP estimates is asymptotically optimal for $n \gg k$:

THEOREM 4.2. *The first order term, as $n \gg k$, of the CV of any (unbiased and nonnegative) linear (adjusted-weights based) estimator of $n_d(v)$ applied to $\mathrm{ADS}(v)$ must be $\geq 1/\sqrt{2k}$.*

PROOF. The inclusion probability of the ith node from v $i \geq k$ in a bottom-k $\mathrm{ADS}(v)$ is $p_i = k/i$. If we had known p_i, the best we could do is use inverse probability weighting, that is, estimate 0 if not sampled and $1/p_i$ if the node is included. The variance

of this ideal estimator is $1/p_i - 1$. There are very weak negative correlations between inclusions of two nodes, making them almost independent (for $i \gg k \gg 1$): The probability p_i given that $j < i$ is included is $\geq (k-1)/i$ and given that j is not included is $k/(i-1)$. The covariance is therefore $O(1/i)$. The sum of all covariances involving node i is therefore $O(1)$ and the sum of all covariances is $O(n)$. The variance of this ideal estimator on a neighborhood of size $n > k$ is at least the sum of variances minus an upper bound on the sum of covariances $\mathsf{Var}[\hat{n}] = \sum_{i=k+1}^{n} \frac{i-k}{k} = \frac{(n+k+1)(n-k)}{2k} - (n-k) - O(n)$. The CV, $\sqrt{\mathsf{Var}[\hat{n}]}/n$, has first order term for $n \gg k$, of $1/\sqrt{2k}$.

Similar arguments apply to k-mins and k-partition ADS. For k-mins ADS, the inclusion probability in $\mathrm{ADS}(v)$ of the ith node from v is $p_i = 1 - (1-1/i)^k \approx k/i$, and we obtain the same sum for $i = 1, \ldots, n$ as with bottom-k ADS. For k-partition, the inclusion probability is $p_i = \mathsf{E}[1/(1+x)]$ where $x \sim B[i, 1/k]$. $\square$

4.4 Permutation estimator

The permutation estimator we present here is applied to a bottom-k ADS that is computed with respect to ranks $\sigma_i \in [n]$ that constitute a random permutation of $[n]$. In terms of information content, permutation ranks dominate random ranks $r(i) \sim U[0,1]$, since random ranks can be associated based on the permutation ranks σ. The main advantage of the permutation estimator is that we obtain tighter estimates when the cardinality we estimate is a good fraction of n. The permutation estimator is only evaluated experimentally.

The permutation estimator, similarly to HIP, is viewed as computed over a stream of elements. In the graph setting, the stream corresponds to scanning of nodes so that first occurrences of nodes are according to increasing distance from v. The entries in $\mathrm{ADS}(v)$ correspond to nodes on which the sketch was updated. A positive weight is then associated with these updates. The weight is an estimate of the number of distinct elements scanned from the previous update (or the beginning if it is the first update) to the current one. We maintain a running estimate $\hat{s}$ on the cardinality s of the set S of distinct elements seen so far. When there is an update, $\hat{s}$ is increased by its weight w.

The first k updates corresponds to the first k distinct elements. Each of these updates has weight 1 and when the cardinality $s \leq k$, our estimate is exact $\hat{s} = s$.

Consider now an update that occur after the first k distinct elements. Let $\mu > k$ be the kth smallest rank in S (which is the kth smallest permutation rank in the bottom-k sketch).

We now argue that after an update, the expected number of distinct nodes until we encounter the next update is $w' = \frac{n-s+1}{\mu-k+1}$. To see this, note that the number of nodes in S with permutation rank μ or below is k. So there are $\mu - k$ remaining nodes with rank smaller than μ amongst those in $[n] \setminus S$. The expectation is that of sampling without replacement until we find a node with permutation rank below μ.

When the update occurs, we would like to compute w' and update our estimate $\hat{s}$. But we actually do not know s. So instead we plug-in the unbiased estimate $\hat{s}$ to obtain $w = \frac{n-\hat{s}+1}{\mu-k+1}$. We then update the bottom-k sketch (and μ if needed) and $\hat{s} \leftarrow \hat{s} + w$.

Note that when $\mu = k$, that is, the k smallest elements of the permutation, those with $\sigma_i \leq k$, are included in S, the probability of an update is 0 as the sketch is saturated. We then need to correct the estimate to account for the number of nodes that are farther than the nodes with permutation rank $[k]$. The correction is computed as follows.

If the cardinality is x, then conditioned on it including all the elements with permutation ranks $[k]$, the expected number of elements that are farther than all the elements with permutation ranks in $[k]$ is $\frac{x-k}{k+1}$. So the expected number of elements till the last update is $x' = x - \frac{n-k}{k+1}$. Note that our estimate $\hat{s}$ was unbiased for x'.

Solving $x - \frac{x-k}{k+1} = x'$ for x we obtain the relation $x = x'\frac{k+1}{k} - 1$. We plug-in $\hat{s}$ for x' and obtain the correction $\hat{x} = \hat{s}\frac{k+1}{k} - 1$. This correction is used when our sketch contains the k elements of permutation ranks $[k]$.

4.5 Simulations

We use simulations to study the Normalized Root Mean Square Error (NRMSE), which corresponds to the CV when estimator is unbiased, and the *Mean Relative Error* (MRE), defined as $\mathsf{E}[|n - \hat{n}|]/n$ of the basic, HIP, and permutation neighborhood-cardinality estimators. We evaluated the basic estimators for all three flavors and the bottom-k HIP estimators. We use sketches with full ranks, because the optimal basic estimators are well understood with full ranks. Actual representation size for "full" ranks is discussed in Section 3.4.

The cardinality $n_d(v)$ is estimated from nodes in $\mathrm{ADS}(v)$ of distance at most d. The structure of the ADS and the behavior of the estimator as a function of the cardinality $n_d(v)$ do not depend on the graph structure. When nodes are presented in increasing distance from v, the ADS only depends on the ranks assigned to these nodes. Our simulation is therefore equivalently performed on a stream of n distinct elements, and ADS content is built from the randomized ranks assigned to these elements. After processing i distinct element, we obtain an estimate of i from the current ADS. We do so for each cardinality. We use multiple runs of the simulation, which are obtained by different randomization of ranks. In case of the permutation estimator, the ranks we use are permutation ranks from a random permutation on all n nodes. For other estimators, the estimate for a certain cardinality does not depend on the total number of nodes.

Figure 2 shows the NRMSE and the MRE estimates by average of multiple simulation runs. We also provide, for reference, the exact values of the CV ($1/\sqrt{k-2}$) and MRE ($\approx \sqrt{2/(\pi(k-2))}$) of the k-mins basic estimator. These values are independent of cardinality and upper bounds the respective measures for the basic bottom-k estimator.

Looking at basic estimators, we can see that (as expected from analysis) for $n \gg k$, the error is similar for all three flavors and the NRMSE is around $1/\sqrt{k-2}$. For smaller values of n, the bottom-k estimator is more accurate than the k-mins which in turn is more accurate than the k-partition estimator: The bottom-k estimator is exact for $k \leq n$ and then the relative error slowly increases until it meets the k-mins error. We can observe that, as explained by analysis, the k-partition estimator is less accurate for $n \leq 2k$.

The figures also include the first-order term (upper bound) for HIP. The results for the bottom-k HIP estimator clearly demonstrate the improvement of the HIP estimators: We can see that the error of the bottom-k HIP estimator is a factor of $\sqrt{2}$ smaller than that of the basic bottom-k estimator. The figures also demonstrate the benefit of using our permutation estimator: The NRMSE and MRE of the permutation estimate were always at most that of HIP. The two are comparable when the estimated cardinality is at most $0.2n$. When it exceeds $0.2n$, we observe a significant advantage for the permutation estimator over plain HIP.

4.6 HIP with base-b ranks

The application and analysis of HIP estimators carries over naturally, retaining unbiasedness even with collisions. Recall that the adjusted weight of i is obtained by first computing a threshold value, based on fixing ranks (and partition) of nodes closer to v

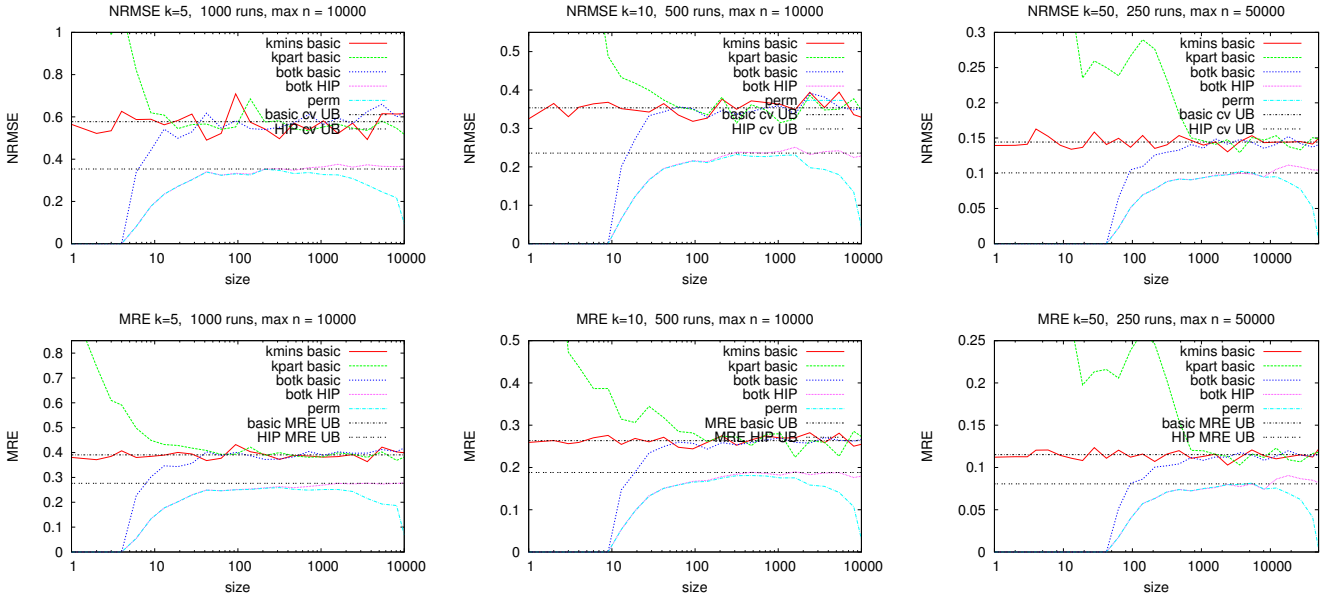

Figure 2: NRMSE (normalized root mean square error) and MRE (mean relative error) of neighborhood size estimators with $k = 5, 10, 50$, as a function of neighborhood size, averaged over multiple runs. We show k-mins, bottom-k, and k-partition basic estimators and our bottom-k HIP and permutation estimators. For reference, we also show the exact values $1/\sqrt{k-2}$ and $1/\sqrt{2(k-1)}$ of the CV of the basic and HIP k-mins estimators. These are upper bounds on the CV of respective bottom-k estimators. We also show $\sqrt{\frac{2}{\pi(k-2)}}$ for the MRE of the basic k-mins estimator and $\sqrt{\frac{1}{\pi(k-1)}}$ as a reference MRE for HIP.

than i. We then use an inverse probability estimate, based on the probability of being below the threshold. The necessary property for unbiasedness of inverse probability estimators, which is satisfied with base-b ranks, is that for any legal threshold τ, there is nonzero probability of having rank that is below τ.

When using base-b ranks, however, the threshold probability p_i we obtain will be "rounded down" from the corresponding full rank probability. Since the probability is strictly smaller than with the full ranks, the threshold inclusion probability τ is lower and therefore the contribution to the variance of the estimate, which is $1/\tau - 1$, is higher. We perform a back-of-the-envelope calculation which shows that τ can be expected to increase by a factor of $\frac{1+b}{2}$, which implies the same-factor increase in variance: Considering a range between discretized values, $a = 1/b^i$ and $ba = 1/b^{i-1}$, and assuming the full rank x lies uniformly in that interval. The full-rank inclusion probability is x whereas the rounded-down one is a. We consider the expectation of the ratio x/a. This expectation is

$$\frac{1}{a(b-1)}\int_a^{ba}\frac{x}{a}dx = \frac{b+1}{2}\ .$$

Simulations in the next section show that this calculation is fairly accurate. We can use this calculation to find a sweet spot for the base b, considering the tradeoff between representation size and variance. The CV as a function of k, b is $\sqrt{\frac{(1+b)}{4(k-1)}}$. The representation size depends on application. If sketch is only used for counting, maintaining few bits for counter, there is diminishing value with smaller bases. If the sketch is used as a sample (which supports selection queries) and stores meta-data (or node IDs), then k is the dominant term and it is beneficial to work with full ranks.

[3] Since both our basic and HIP estimators have CV of the order of $1/\sqrt{k}$, we can see that with $b = \log k$ precision bits, little accuracy is lost.

[3] For permutation position (see Appendex 4.4), we consider $\lceil \log_2 n \rceil$ bit representation with the exponent being the number of leading zeros and significant bits follow. The exponent of a minimum rank value in a set of n nodes has expected size $\log\log n$.

5. APPROXIMATE DISTINCT COUNTING

Our HIP estimators (with full or base-b ranks) can be used to approximate the number of distinct elements in a stream. A Min-Hash sketch is maintained for the distinct elements on the prefix of the stream that is processed.

To apply HIP, we augment the Min-Hash sketch with an additional register which maintains an approximate count of the number of distinct elements. Each time the sketch is updated, we compute the adjusted weight of the element and accordingly increase the count by that amount. Since the expected total number of updates is $\leq k \ln n$, where n is the number of distinct elements in the stream (see Lemma 2.2), the additional work performed for an update balances out as a diminishing fraction of the total stream computation.

An explicit representation of the additional counter as an approximate counter (see Appendix A) would require storing the exponent, which is of size $\lceil \log\log n \rceil + 1$ and $\lceil \log_2 \sqrt{(4k/3)} + 4 \rceil$ significant bits (precision with respect to the CV of HIP). The exponent can more efficiently be stored as an offset to the exponent values stored in the sketch, removing its dependence on n. Thus, using only $O(\log_2 k)$ for the approximate count. An even more compact representation of the approximate count also eliminates the dependence on k, and requires only few bits in total. To do that we represent the HIP estimate as a correction of a basic estimate obtained from the Min-Hash sketch. The correction can be expressed as a signed multiplier of $\hat{n}/\sqrt{k}$ using a fixed number of bits. When the sketch is updated, we recompute the basic estimate

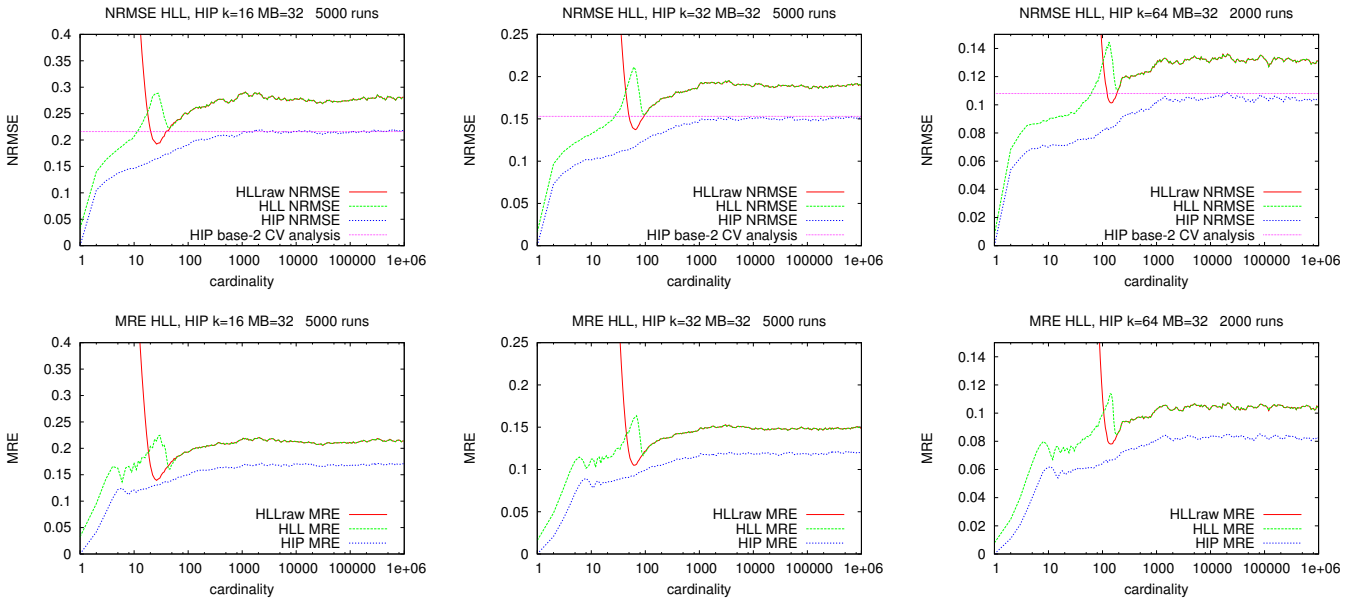

Figure 3: HIP and HLL raw and bias-corrected estimators. Applied with $k = 16, 32, 64$ and 5-bit counters. (k-partition base-2 Min-Hash sketches)

$\hat{n}$ and accordingly update the correction to be with respect to the new HIP estimate.

HIP is very flexible. It applies to all three sketch flavors, to full and to base-b ranks, and also works with truncated registers that can get saturated. In this case we simply take the update probability of a saturated register to be 0. The HIP estimate quality gracefully degrades with the number of saturated registers. Eventually, if all registers are saturated, the HIP estimate saturates and becomes biased. In order to compare the HIP estimate with HyperLogLog (HLL) [28, 30], which is the state of the art approximate distinct counter, we implemented it on the same Min-Hash sketch that is used by HLL. HLL uses k-partition Min-Hash sketches with base-2 ranks. The registers have 5 bits and are thus saturated at 31. Pseudo code for the HIP estimator when applied to the HLL sketch is provided in Figure 4.

Require: Random uniform hash functions: $\text{BUCKET}(v) : [k]$, $r(v)$: first 32 bits of $U[0,1]$.

Initialization:
for $i = 1, \ldots, k$ **do** $M[i] \leftarrow 0$ ▷ $M[i]$ are 5-bit registers
$c \leftarrow 0$ ▷ c is an approximate counter

Processing stream element v**:**
$h(v) \leftarrow \min\{31, \lceil -\log_2 r(v) \rceil\}$
if $h(v) > M[\text{BUCKET}(v)]$ **then**
    $c \leftarrow c + \left(\sum_{i=1}^{k} I_{M[i]<31} 2^{-M[i]}\right)^{-1}$
    $M[\text{BUCKET}(v)] \leftarrow h(v)$

Figure 4: Pseudo code for HIP on the HyperLogLog Min-Hash sketches: k-partition, base-2, each register uses 5 bits. To apply HIP we maintain an additional register c.

Figure 3 shows results for the performance of the HIP and HLL estimators. Noting again that each simulation can be performed on any stream of distinct elements (multiple occurrences do not update the sketch or the estimate). We implemented HyperLogLog using the pseudocode provided in [28]. We show both the raw estimate and the improved bias corrected estimate as presented. The Figure also shows the back-of-the-envelope approximate bound we calculated for the CV of HIP, $\sqrt{\frac{b+1}{4(k-1)}}$, and we can see that it approximately matches simulation results.

A more recent and more complicated implementation of HLL [30] obtains improved performance. The improvement amounts to smoothing out the "bump" due to the somewhat ad-hoc bias reducing component, but the asymptotic behavior is the same as the original hyperLogLog. We can see that HIP obtains an asymptotic improvement over HLL and also has a smooth behavior. Moreover, HIP is unbiased (unless all counters are saturated) and elegant, and does not require corrections and patches as with [28, 30].

We quantify the improvement more precisely in terms of the number k of registers: The NRMSE of HLL is $\approx 1.08/\sqrt{k}$ versus $\approx \sqrt{3/(4k)} \approx 0.866/\sqrt{k}$ of HIP. This means that an HLL estimator requires $\approx 0.56k$ more registers for the same square error as a HIP estimator. As discussed above, HIP requires an additional register c, but its benefit, in terms of accuracy outweighs the overhead.

Some encoding optimizations that were proposed for HLL [30] and elsewhere [32] can also be integrated with HIP. In particular, the content of the k registers is highly correlated can be represented compactly by storing only one value and offsets for others (the expected size of each offset is constant). Recall that the "exponent" component of the approximate count c can also be represented as an offset (see discussion above).

We note that HIP permits us to work with a different base, and get further improvements with respect to HyperLogLog. Consider using base $b = 2^{1/i}$ for $i \geq 1$. With smaller base, we need larger counters but we also have a smaller variance. We need about $\log_2 \log_b n$ bits per register, for counting up to cardinality (number of distinct elements) $n/16$ (since we want to have the counters large enough so that at most a fraction of them get saturated). Since

$\log_2 \log_b n = \log_2(\log_2 n/\log_2 b) = \log_2 \log_2 n - \log_2 \log_2 b \approx \log_2 \log_2 n + \log_2 i$, it means we need about $\log_2 i$ additional bits per register relative to base-2. The CV is $\approx \sqrt{\frac{b+1}{4(k-1)}}$. So with $i = 1$ (base $b = 2$) we had CV $\approx 0.866/\sqrt{k}$, with $i = 2$ ($b = \sqrt{2}$), we need 1 additional bit per register but the CV is $\approx 0.777/\sqrt{k}$, meaning that we need 20% fewer registers for the same error as when using base-2. The advantage of base-$\sqrt{2}$ kicks in when n exceeds about 3×10^8. If the counting algorithm also retains a sample of distinct elements (such as with reservoir sampling) and thus IDs of sampled elements are retained, representation size is dominated by $k \log n$ in which case we might as well use full ranks for the approximate distinct count.

Our evaluation aims at practice and we mention some differences with the theory literature. First, our analysis applies to random hash functions. This is justified by simulation results with standard generators. We mention that a lower bound of Alon et al. [3] on the sketch size has logarithmic dependence on the cardinality (and there is a matching upper bound by Kane et al. [32]) whereas the HLL sketch has a much smaller, double logarithmic, size. The reason is that the lower bound "includes" the encoding of the hash function as part of the sketch, a requirement which is not justified when many counters use the same hash function.

Lastly, we comment on the mergeability of our extended Min-Hash sketches. Mergeability means that we can obtain a sketch of the union of (overlapping) data sets from the sketches of the sets. This property is important when parallelizing or distributing the computation. The Min-Hash component of the extended sketch are mergeable, but to correctly merge the counts, we need to estimate the overlap between the sets. This can be done using the similarity estimation hat of Min-Hash sketches. We leave further details for future work.

6. DISTANCE RANGES

We can obtain a small relative error (CV of $1/\sqrt{2(k-1)}$ with HIP) for estimating neighborhood cardinalities, that is, $n_d(v)$ for any v and d. In this section we explore the estimation quality For $n_J(v)$, where J is an arbitrary interval J, and in particular for estimating the number of nodes within distance exactly d from v. These problems generalize diameter estimation. The best known approximation of the diameter D that can be obtained in near linear time is $[0.5D, D]$: we pick an arbitrary node v and perform a single source shortest paths computation to find the farthest node from v. When the graph is undirected, the distance to the farthest node is at least $0.5D$. It is widely believed that a better approximation requires polynomially more time.

Consider estimating $n_{(a,b]}(v)$ from $\mathrm{ADS}(v)$ using the HIP estimator $\sum_{j \in N_{(a,b)}(v)} a_{vj}$. For a node j, the probability that $j \in \mathrm{ADS}(v)$ is $\min\{1, k/\pi_{vj}\}$. The expected number of nodes from $N_{(a,b]}(v)$ in $\mathrm{ADS}(v)$ is $k(H_{n_b(v)} - H_{n_a(v)}) \approx k \ln(n_b(v)/n_a(v))$. Since inclusions are negatively correlated, the variance of the HIP estimate is $\leq \sum_{i=n_a(v)+1}^{n_b(v)} \frac{i-1}{k-1} = \frac{1}{k-1} \frac{(n_b(v)+n_a(v)+1)(n_b(v)-n_a(v))}{2}$ and the CV, which is the ratio of the square-root of the variance to the mean $(n_b(v) + n_a(v))$, is $\leq \sqrt{\frac{1}{k-1} \frac{n_b(v)+n_a(v)}{2(n_b(v)-n_a(v))}}$. We can see that when $n_b(v) \gg (1 + 1/k)n_a(v)$, that is, when there is sufficient expansion, we obtain a vanishing CV with k.

For undirected graphs, we are able to provide bounds that do not depend on expansion: We show that when there is a pair with distance d in the graph, then with high probability there must be a sampled pair, that is a pair i, j such that $i \in \mathrm{ADS}(j)$ or vice versa, so that $d_{ij} \in J = [d/2, 3d/2]$. Moreover, for any i, j of distance d we can estimate with good relative error the number of pairs of nodes $n_J(i) + n_J(j)$ that are within distance that is in J from either i or j. These bounds match and extend what we can do for the diameter in near-linear time.

THEOREM 6.1. *For an undirected graph, for all i, j, and $J = [d_{ij}/2, 3d_{ij}/2]$, the HIP estimator $\hat{n}_J(i)+\hat{n}_J(j)$ has CV $\leq \sqrt{\frac{2}{k-1}}$.*

PROOF. Let $d \equiv d_{ij}$ and $J = [d/2, 3d/2]$. Clearly, $N_{(0,d/2)}(i)$ and $N_{(0,d/2)}(j)$ are disjoint, $N_{(0,d/2)}(i) \subset N_J(j)$, and vice versa.

We first consider the covariances of the adjusted weight a_{jh} (for $h \in N_J(j)$) and $a_{i\ell}$ (for $\ell \in N_J(i)$). The covariance $\mathsf{Cov}[a_{hj}, a_{\ell i}]$ is not zero only when $\ell = h$. In this case, there is positive correlation and $\mathsf{Cov}[a_{jh}, a_{ih}] \leq \max\{\mathsf{Var}[a_{ih}], \mathsf{Var}[a_{jh}]\}$.

As a coarse upper bound,

$$\mathsf{Var}[\hat{n}_J(i) + \hat{n}_J(j)] \leq 2\Big(\sum_{h|d_{ih} \in J} \mathsf{Var}[a_{ih}] + \sum_{\ell|d_{j\ell} \in J} \mathsf{Var}[a_{j\ell}]\Big) . \tag{9}$$

We have

$$\begin{aligned}
\sum_{h|d_{ih} \in J} \mathsf{Var}[a_{hi}] &= \sum_{h=n_{(0,d/2)+1}(i)}^{n_{(0,3d/2]}(i)} \frac{h-1}{k-1} \\
&= \frac{1}{2(k-1)}(n_{(0,3d/2]}(i) + n_{(0,d/2]}(i) - 2)n_J(i) \\
&\leq \frac{1}{2(k-1)}(n_J(i) + 2n_{(0,d/2]}(i))n_J(i) \\
&\leq \frac{1}{2(k-1)}(n_J(i) + 2n_J(j))n_J(i) = \frac{n_J(i)^2 + 2n_J(i)n_J(j)}{2(k-1)} .
\end{aligned}$$

The last inequality uses the fact that $N_{(0,d/2]}(i) \subset N_J(j)$ and therefore $n_{(0,d/2]}(i) \leq n_J(j)$. Substituting in (9) we obtain

$$\begin{aligned}
\frac{\mathsf{Var}[\hat{n}_J(i) + \hat{n}_J(j)]}{(n_J(i) + n_J(j))^2} &\leq \frac{1}{k-1} \frac{n_J(i)^2 + 4n_J(i)n_J(j) + n_J(j)^2}{(n_J(i) + n_J(j))^2} \\
&= \frac{1}{k-1} \frac{(n_J(i) + n_J(j))^2 + 2n_J(i)n_J(j)}{(n_J(i) + n_J(j))^2} \\
&\leq \frac{1.5}{k-1}
\end{aligned}$$

□

Conclusion

ADSs, introduced two decades ago, are emerging as a powerful tool for scalable analysis of massive graphs. We introduce HIP estimators, which apply to an extensive class of natural queries, are simple to apply with all sketch flavors, and significantly improve over state of the art. For neighborhood cardinalities and closeness centralities, HIP estimators have at most half the variance of previous estimators. Moreover, HIP estimators outperform state of the art practical estimators for approximate distinct counting on data streams.

In follow-up work on social network analysis, we applied HIP for ADS-based estimation of closeness similarity of two nodes [15] and timed-influence of a set of seed nodes [16].

We provided here a unified view of ADS flavors and algorithms which we hope will facilitate further applications of these versatile structures. Beyond the applications mentioned already, an ADS set can be used as a spanner or emulator. We recently showed that when ADSs are used as distance oracles [15], they have the worst-case guarantees of [45] on undirected graphs and a very good performance in practice on both directed and undirected graphs.

Lastly, we obtained interesting insights on Min-Hash sketch-based cardinality estimation through rather straightforward applications of the classic theory of point estimation. Specifically, we obtain exact (rather than asymptotic) lower bounds on the variance of an estimator applied to a certain sketch. We expect that this powerful theory, as is, or with some adaptations to discrete settings, can provide further insights on other sketch structures.

Acknowledgement

The author would like to thank Seba Vigna for helpful pointers.

7. REFERENCES

[1] D. Aingworth, C. Chekuri, P Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999.

[2] T. Akiba, Y. Iwata, and Y. Yoshida. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *SIGMOD*, pages 349–360, 2013.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. System Sci.*, 58:137–147, 1999.

[4] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. In *WebSci*, pages 33–42, 2012.

[5] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM*. ACM, 2002.

[6] A. Bavelas. A mathematical model for small group structures. *Human Organization*, 7:16–30, 1948.

[7] D. Blackwell. Conditional expectation and unbiased sequential estimation. *Annals of Mathematical Statistics*, 18(1), 1947.

[8] P. Boldi, M. Rosa, and S. Vigna. HyperANF: Approximating the neighbourhood function of very large graphs on a budget. In *WWW*, 2011.

[9] P. Boldi, M. Rosa, and S. Vigna. Robustness of social networks: Comparative results based on distance distributions. In *SocInfo*, pages 8–21, 2011.

[10] P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 2014.

[11] K. R. W. Brewer, L. J. Early, and S. F. Joyce. Selecting several samples from a single population. *Australian Journal of Statistics*, 14(3):231–239, 1972.

[12] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29. IEEE, 1997.

[13] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Proc.of the 11th Annual Symposium on Combinatorial Pattern Matching*, volume 1848 of *LNCS*, pages 1–10. Springer, 2000.

[14] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. System Sci.*, 55:441–453, 1997.

[15] E. Cohen, D. Delling, F. Fuchs, A. Goldberg, M. Goldszmidt, and R. Werneck. Scalable similarity estimation in social networks: Closeness, node labels, and random edge lengths. In *COSN*, 2013.

[16] E. Cohen, D. Delling, T. Pajor, and R. Werneck. Influence computation scaled-up in sketch space, 2014. Manuscript.

[17] E. Cohen and H. Kaplan. Spatially-decaying aggregation over a network: model and algorithms. *J. Comput. System Sci.*, 73:265–288, 2007. Full version of a SIGMOD 2004 paper.

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

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

[20] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. *J. Algorithms*, 59:19–36, 2006.

[21] P. Crescenzi, R. Grossi, L. Lanzi, and A. Marino. A comparison of three algorithms for approximating the distance distribution in real-world graphs. In *TAPAS*, 2011.

[22] Ch. Dangalchev. Residual closeness in networks. *Phisica A*, 365, 2006.

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

[24] M. Durand and P. Flajolet. Loglog counting of large cardinalities (extended abstract). In *ESA*, 2003.

[25] D. Eppstein and J. Wang. Fast approximation of centrality. In *SODA*, pages 228–229, 2001.

[26] W. Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, New York, 1971.

[27] P. Flajolet. Approximate counting: A detailed analysis. *BIT*, 25, 1985.

[28] P. Flajolet, E. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *Analysis of Algorithms (AOFA)*, 2007.

[29] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. System Sci.*, 31:182–209, 1985.

[30] S. Heule, M. Nunkesser, and A. Hall. HyperLogLog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *EDBT*, 2013.

[31] 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.

[32] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, 2010.

[33] E. L. Lehmann and H. Scheffé. Completeness, similar regions, and unbiased estimation. *Sankhya*, 10(4), 1950.

[34] P. Li, , K. W. Church, and T. Hastie. One sketch for all: Theory and application of conditional random sampling. In *NIPS*, 2008.

[35] P. Li, A. B. Owen, and C-H Zhang. One permutation hashing. In *NIPS*, 2012.

[36] M. H. Malewicz, G.and Austern, A.J.C Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD*. ACM, 2010.

[37] R. Morris. Counting large numbers of events in small registers. *Comm. ACM*, 21, 1977.

[38] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi. Naiad: a timely dataflow system. In *SOSP*, 2013.

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

[40] T. Opsahl. Closeness centrality in networks with disconnected components. `http://toreopsahl.com/2010/03/20/`, 2010.

[41] C. R. Palmer, P. B. Gibbons, and C. Faloutsos. ANF: a fast and scalable tool for data mining in massive graphs. In *KDD*, 2002.

[42] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC*. ACM, 2013.

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

[44] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832, 1956.

[45] M. Thorup and U. Zwick. Approximate distance oracles. In *Proceedings of the 33th Annual ACM Symposium on Theory of Computing, Crete, Greece*, pages 183–192, 2001.

APPENDIX

A. APPROXIMATE COUNTING

We provide some details here on approximate (not distinct!) counters. We used such counters in our implementation of approximate distinct counters.

An approximate counter is applied to a stream of positive integers $\{w_i\}$ and represents $n = \sum_i w_i$ approximately. Whereas an exact representation takes $\lceil \log_2 n \rceil$ bits, an approximate counter that uses only $O(\log \log n)$ bits was proposed by Morris [37] and analysed and extended by Flajolet [27]. This *Morris counter* is an integer $x \geq 0$ and the estimate is $\hat{n} = b^x - 1$, where the (fixed) base $b > 1$ controls a tradeoff between approximation quality and representation size.

Since originally presented only for increments of 1, we provide procedures here for efficient weighted updates and for merges of two counters. An update of Y to a counter x is performed as follows: Let $i \leftarrow \lfloor \log_b(Y/b^x - 1) \rfloor$ be the maximum such that increasing the counter by i would increase the estimate by at most Y. We then compute the leftover $\Delta \leftarrow Y - b^x(b^i - 1)$ and update $x \leftarrow x + i$. Lastly, we increase x by 1 with probability $\Delta/(b^x(b-1))$ (this is an inverse probability estimate of Δ). Merge of two Morris counters x_1, x_2 is handled the same as incrementing x_1 with $b^{x_2} - 1$. The estimator $\hat{n}$ is clearly unbiased (by induction on updates).

It is easy to show that the variance is dominated by the analysis in [37, 27] for increments, since it can only improve when two consecutive updates are combined to a single update with the sum of their values. Intuitively, only the "leftover" part of the updates contributes to the variance at all.

This is particularly relevant to us since we use an approximate counter in our HIP approximate distinct counter. The update magnitudes are increasing and typically are about $1/k$ of the total, therefore, with the choice of $b \leq 1 + 1/k$, the variance is significantly lower than in [37, 27]. The number of bits needed for counter representation is $\log_2 \log_b n \approx \log_2 \log_2 n + \log_2(1/(b-1))$ and the CV is about $(b-1)$. When using $b = 1 + 1/2^j$ we obtain that with j additional bits in the representation we can obtain relative error of $1/2^j$.

B. NON-UNIFORM NODE WEIGHTS

To simplify the presentation, we focused on uniform weights but briefly discuss the extension to arbitrary weights, which are appropriate in many applications. The closeness centrality definition (2) incorporates node weights $\beta(j)$. We can also consider neighborhood weights

$$n_d(v) = \sum_{j | d_{vj} \leq d} \beta(j)$$

instead of just cardinalities.

To obtain the same CV as estimators for the uniform weights, we need to compute the ADSs with respect to the weights $\beta(i)$. To do that we draw the rank $r(i)$ for node i using the exponential distribution with parameter $\beta(i)$ [14, 18]. This is the same as drawing uniform ranks $r'(i)$ and using ranks $r(i) = -\ln(1 - r'(i))/\beta(i)$. With these ranks, nodes with higher β values have higher inclusion probabilities. The same ADS definitions and algorithms apply, simply using the modified ranks. Note however, that ADSs can have larger expected sizes (the β weights can be viewed as emulating multiple copies of a node).

We first discuss Min-Hash cardinality estimators. The k-mins basic estimator (with exponentially distributed ranks) applies with same CV of $1/\sqrt{k-2}$ to weighted k-mins Min-Hash sketches [14]. An estimator for weighted bottom-k Min-Hash sketches was given in [19] for weighted sampling without replacement and general order samples. An alternative with bottom-k is to use $r(i) = r'(i)/\beta(i)$, which corresponds to Sequential Poisson (Priority) sampling [39, 23]. The priority sampling estimator [23] applies to the Min-Hash sketch.

The HIP estimators we presented here naturally extend, and remain unbiased, with any weight-based rankings $r(i) = f(i, \beta(i), r'(i))$. We simply compute the probability that $r(i)$ is below the threshold value. If $r(i)$ are exponentially distributed, the CV of estimating neighborhood weights and centralities is at most $1/\sqrt{2(k-1)}$.

C. ESTIMATOR WHICH USES ONLY THE ADS SIZE

We derive the *size* estimator which is the unique unbiased cardinality estimator that is only based on the size (number of entries) of the ADS. Specifically, to estimate the cardinality of $N_d(v)$ we look at the number of entries in $\mathrm{ADS}(v)$ with distances at most d. In a stream context, we can apply the size estimator to the number of updates (which resulted in modifying) the Min-Hash sketch. The size estimator is weaker than the HIP estimator but uses less information. This estimator is of interest in a setting where one can observe the approximate counter as a black box, only observing the number of modifications.

The estimator we derive below is applied to the *number of entries* in a bottom-k ADS that are within distance at most d from v. The estimator assumes that the ADS is computed with respect to "unique" distances. That is, we apply some symmetry breaking and ADS may include multiple nodes of same distance.

LEMMA C.1. *The unique unbiased estimator E_s of $|N_d(v)|$ based solely on ADS size $s = |N_d(v) \cap \mathrm{ADS}(v)|$ is*

$$E_s = \begin{cases} s \leq k \quad : & s \\ \textit{otherwise} \quad : & k(1 + \frac{1}{k})^{s-k+1} - 1 \,. \end{cases}$$

PROOF. Let $C_{i,\ell}$ be the probability that exactly i nodes are sampled from the first ℓ. For $\ell \geq k$ and $i < k$ or for $\ell \leq k$ and $i < \ell$, $C_{i,\ell} = 0$. If $\ell \leq k$, then $C_{\ell,\ell} = 1$. We have the relations

$$\begin{aligned} \ell > k: \quad & C_{\ell,\ell} = (k/\ell) C_{\ell-1,\ell-1} \\ k < i < \ell: \quad & C_{i,\ell} = (1 - k/\ell) C_{i,\ell-1} + (k/\ell) C_{i-1,\ell-1} \\ k < \ell: \quad & C_{k,\ell} = (1 - k/\ell) C_{k,\ell-1} \end{aligned}$$

(k/ℓ) is the probability that the ℓth node is one of the first k in the random permutation induced on the ℓ nodes closest to v.

If $s < k$, which is only possible if $n_r(v) = s$, we have $E_s = s$. If $s = k$, to be unbiased for the case where $n_r(v) = k$ and this is the only possible count, we have $E_k = k$. Otherwise, for $s > k$, we have that any estimator that is unbiased on neighborhoods of size s must satisfy $s = \sum_{i=k}^{s} E_i C_{i,s}$, which we rearrange to obtain

$$E_s = \frac{s - \sum_{i=k}^{s-1} E_i C_{i,s}}{C_{s,s}} \,. \tag{10}$$

We iteratively apply (10) to uniquely determine E_s for $s \geq k+1$. To determine E_{k+1}, we consider the two possible ADS counts are k and $k+1$ with respective probabilities $C_{k,k+1} = 1/(k+1)$ and $C_{k+1,k+1} = k/(k+1)$. From (10) $k + 1 = E_k C_{k,k+1} + E_{k+1} C_{k+1,k+1} = k/(k+1) + E_{k+1} k/(k+1)$. We obtain $E_{k+1} = (k+1)^2/k - 1$. It can be verified that the general solution satisfies $E_s = k(1 + \frac{1}{k})^{s-k+1} - 1$. $\square$

This estimator is also applicable with $k = 1$, in which case it is simply 2^s.

D. ADS WITHOUT TIE BREAKING

We provide here a modified ADS definition, and respective HIP probabilities, for when there is a smaller set of distinct distances. The advantages of the modified definition is a smaller ADS size: The modified ADS (we provide here the bottom-k flavor) includes a subset of the entries that would have been included under the original definition (with tie breaking on distances), but at most k entries (those with smallest ranks) from each distinct distance.

Formally, a node u with is included in (modified) $\mathrm{ADS}(v)$ if $r(u)$ is smaller than the kth lowest rank amongst nodes within distance *at most* d_{vu} from v.

We can assign HIP inclusion probabilities for the modified ADS as follows.

For each v, u, we compute the probability of u, conditioned on fixed ranks of all other nodes excluding u, of u having one of the $k-1$ smallest ranks amongst nodes with distance in d_{vu} from v. We compute this probability only for nodes that satisfy this condition of having one of these $k-1$ smallest ranks. The threshold probability is that kth smallest rank. Note that a node $u \in \mathrm{ADS}(v)$ that has the kth smallest rank in $N_{d_{vu}}(u)$ is not considered "sampled."

The modified HIP probabilities can be applied to the same queries. The HIP probabilities of an entry in the modified ADS are at most the values in the full with tie-breaking ADS. Therefore, adjusted weights and variances are higher. The CV is at most $1/\sqrt{k-2}$, this is because when all distances are the same (say edges have 0 lengths and the ADS is a reachability sketch), the modified ADS is a bottom-k Min-Hash sketch of the reachability set.

E. ADS COMPUTATION

We provide a unified presentation of ADS algorithms and propose some extensions. There are two existing approaches. The first, PRUNED DIJKSTRA'S is based on pruned applications of Dijkstra's single-source shortest paths algorithm (BFS when unweighted) [14, 18]. A pseudo-code is provided in Algorithm 1. The second, DP, is applicable only to unweighted graphs and based on dynamic programming or Bellman-Ford shortest paths computation. It was (implicit) in [41, 8].

PRUNED DIJKSTRA'S was first proposed for k-mins sketches [14] and extended to bottom-k sketches in [18]. DP was considered with k-mins [41] and k-partition sketches [8]. Both approaches, however, can be easily adopted to work with all three ADS flavors.

LOCAL UPDATES, proposed here and provided in pseudo-code as Algorithm 2, extends DP to weighted graphs. Local updates has a simple node-centric form, which is appropriate for MapReduce or similar platforms [36, 38]. When the operations are synchronized, as with MapReduce, the total number of iterations needed until no more updates can be performed is bounded by the diameter of the graph (maximum over pairs of nodes of the number of hops in the shortest path between them).

Algorithm 1 ADS set for G via PRUNED DIJKSTRA

for u by increasing $r(u)$ **do**
  Perform a pruned Dijkstra from u on G^T (the transpose graph)
  When visiting node v:
  if $|\{(x,y) \in \mathrm{ADS}(u) \mid y < d_{vu}\}|$ **then** prune Dijkstra at v
  else $\mathrm{ADS}(v) \leftarrow \mathrm{ADS}(v) \cup \{(r(u), d_{vu})\}$

Both PRUNED DIJKSTRA'S and DP can be performed in $O(km \log n)$ time (on unweighted graphs) on a single-processor in main memory, where n and m are the number of nodes and edges in the graph.

These algorithms maintain a partial ADS for each node, as entries of node ID and distance pairs. $\mathrm{ADS}(i)$ is initialized with the pair $(i, 0)$. The basic operation we use is *edge relaxation* (named after the corresponding operation in shortest paths computations). When relaxing (i, j), $\mathrm{ADS}(i)$ is updated using $\mathrm{ADS}(j)$. For bottom-k, the relaxation modifies $\mathrm{ADS}(i)$ when $\mathrm{ADS}(j)$ contains a node v such that $r(v)$ is smaller than the kth smallest rank amongst nodes in $\mathrm{ADS}(i)$ with distance at most $d_{jv} + w_{ij}$ from i. More precisely, if v was inserted to $\mathrm{ADS}(j)$ after the most recent relaxation of the edge (i, j), we can update $\mathrm{ADS}(i)$ using INSERT$(i, v, d_{jv} + w_{ij})$:

1: **function** INSERT(i, x, a) ▷ update $\mathrm{ADS}(i)$ with (x, a)
2: **if** $x \notin \mathrm{ADS}(i)$ **then** ▷ if $x \in \mathrm{ADS}(i)$ do nothing.
3: **if** $r(x) < \mathrm{k^{th}}\{r(y) | y \in \mathrm{ADS}(i) \wedge d_{iy} \leq a\}$ **then**
4: $\mathrm{ADS}(i) \leftarrow \mathrm{ADS}(i) \cup \{(x, a)\}$

Both PRUNED DIJKSTRA'S and DP perform relaxations in an order which guarantees that inserted entries are part of the final ADS, that is, there are no other nodes that are both closer and have lower rank: PRUNED DIJKSTRA'S iterates over all nodes in increasing rank, runs Dijkstra's algorithm from the node on the transpose graph, and prunes at nodes when the ADS is not updated. DP performs iterations, where in each iteration, all edges (i, j) such that $\mathrm{ADS}(j)$ was updated in the previous step, are relaxed. Therefore, entries are inserted by increasing distance.

The edge relaxation function INSERT is stated so that it applies for both algorithms, but some of the conditional statements are redundant: The test $x \notin \mathrm{ADS}(i)$ is not needed with PRUNED DIJKSTRA'S (we only need to record that if/when node i was already updated in the current Dijkstra) and the test $d_{iy} < a$ is not needed with DP (since all entries in current iteration are of distance at most a).

To obtain a bound on the number of edge relaxations performed we note that a relaxation of (i, j) can be useful only when $\mathrm{ADS}(j)$ was modified since the previous relaxation of (i, j). Therefore, each relaxation can be "charged" to a modification at its sink node, meaning that the total number of relaxations with sink j is bounded by the size of $\mathrm{ADS}(j)$ times the in-degree of j. We obtain that the expected total number of relaxations is $O(km \log n)$.

We provide details on bottom-k ADS algorithms. A k-mins ADS set can be computed by performing k separate computations of a bottom-1 ADS sets (using k different permutations). To compute a k-partition ADS set, we perform a separate bottom-1 ADS computation for each of the k buckets (but with the modification that the ADS of nodes not included in the bucket is initialized to $\emptyset$). The total number of relaxations is $O(km \log n)$, which again is m times the expected size of of k-partition ADS (which is the same as the size of a bottom-k ADS).

LOCAL UPDATES incurs more overhead than PRUNED DIJKSTRA, as entries can also be deleted from the ADS (in a CLEANUP step). For adversarially constructed graphs (where distance is inversely correlated with hops), the overhead can be made large. In practice, however, we can expect a small overhead. We can also guarantee an $O(\log n)$ overhead by settling for $(1+\epsilon)$-approximate ADSs (where $\epsilon > 1/n^c$). A $(1+\epsilon)$-approximate $\mathrm{ADS}(u)$ satisfies

$$v \notin \mathrm{ADS}(u) \implies r(v) > \mathrm{k}^{\mathrm{th}}_x\{(x,y) \in \mathrm{ADS}(u) \mid y < (1+\epsilon)d_{uv}\}\,.$$

We can compute a $(1+\epsilon)$-approximate ADS set by updating the ADS only when on updates INSERT(i, x, a) for which the condition is not violated, that is,

$$r(x) < \mathrm{k^{th}}\{r(y) | y \in \mathrm{ADS}(i) \wedge d_{iy} \leq a(1+\epsilon)\}\ .$$

It is not hard to show that with this restriction to approximate ADS, the overhead on the total number of updates is bounded by $\log_{1+\epsilon} \frac{nw_{\max}}{w_{\min}}$,

where $w_{\max}$ and $w_{\min}$ are the largest and smallest edge lengths. With ϵ at least polynomially small, we can assume wlog that the ratio $w_{\max}/w_{\min}$ is polynomial. Obtaining a logarithmic overhead.

Algorithm 2 ADS set for G via LOCAL UPDATES

Initialize:
for u **do** $\mathrm{ADS}(u) \leftarrow \{(r(u), 0)\}$

Send updates:
if (r, d) was added to $\mathrm{ADS}(u)$ in the previous iteration **then** $\forall\{y \mid (u, y) \in G\}$ send $(r, d + w(u, y))$ to y.

Process updates:
if node u received (r, d) **then**
  if $r < \mathrm{k}^{\mathrm{th}}_x\{(x, y) \in \mathrm{ADS}(u) \mid y < d\}$ **then**
    $\mathrm{ADS}(u) \leftarrow \mathrm{ADS}(u) \cup \{(r(v), d)\}$
    CLEAN-UP $\mathrm{ADS}(v)$ ▷ Scan entries (x, y) such that $y > d$ by increasing y. Remove (x, y) if $x > \mathrm{k}^{\mathrm{th}}_h\{(h, z) \in \mathrm{ADS}(u) \mid z < y\}$

In the sequel, we discuss various additional aspects of ADS computation.

Limited ADS computation.

When memory is constrained, we can benefit when not maintaining the full ADS in "active" memory, but only maintaining the *threshold* information required to proceed with the computation. We refer to this as a *limited* ADS computation.

With PRUNED DIJKSTRA'S, ranks are processed in increasing order. In the iteration from node i, when visiting a node, we need to have access to all rank-distance pairs in the ADS constructed so far at the visited node. With DP, processing is in increasing distance. To determine if a proposed entry indeed contributes to the ADS, we only need to maintain the Min-Hash sketch of ranks presented so far, which has size k. The ANF [41, 21] and hyperANF [8] algorithms are essentially limited DP computation with base-2 ranks. Streaming approximate distinct count estimators [29, 24] were applied to the base-2 Min-Hash sketch of the current neighborhood to estimate its size after each DP iteration. The results from different nodes were aggregated after each iteration to produce an estimate of the total number of pairs within each distance.

PRUNED DIJKSTRA'S base-b: When we work with base-b ranks, which are not uniquely assigned for nodes, we have to ensure that the ADS is not updated twice in the same iteration. This can be done by marking each node after the first visit in each iteration and stopping the search on subsequent visits. Note that the threshold ADS at each node can include multiple occurrences of the same base-b rank value (each corresponding to a distinct node). But because iteration order corresponds to the order on the full rank, the entries correspond to the entries of the full-rank ADS. To obtain the explicit ADS (with node IDs), we can export each new entry $i \in \mathrm{ADS}(v)$ and the distance d_{iv} to a slower medium. After the computation we can aggregate all entries of $\mathrm{ADS}(v)$ for each v.

With DP, base-b, and bottom-k sketches, we must treat rank values in the same ADS as unique. This is needed to avoid having the same node contribute to multiple entries in an ADS of another node.

Cost of relaxations.

The expected total number of relaxations is $O(mk \log n)$ but the expected number of relaxations that actually result in an update is $O(nk \log n)$. This distinction is important because relaxations which result in an update are more costly.

We first consider relaxations with DP. With k-mins and k-partition ADS, we can retain with each update the index (out of k) which was modified since the last update. If we do so, then the cost of relaxing an edge is $O(1)$, since we only need to look at the rank value in the modified index. If the index is not retained, we can perform a coordinate wise minimum of the k entries, in time $O(k)$. The better choice depends on the hardware.

With bottom-k ADS, we maintain the current bottom-k ranks in active memory. When relaxing an edge we compare the newly inserted rank value in the sink ADS. If the entries are maintained in a max-heap, the maximum entry is compared with the new value. If the new value is smaller, it is inserted into the heap and the max entry is removed. The cost is $O(1)$ if the ADS is not updated (node is not inserted) but $O(\log k)$ otherwise (the max node is removed from the heap and the new node is inserted). Alternatively, we can maintain the k values in a sorted list and each update takes $O(k)$ time.

Relaxations with PRUNED DIJKSTRA'S are less efficient than with DP as we need to search for the minimum rank in a distance range which increases update times by a factor of $\log k$.

Removing unique distances assumption in analysis.

The strict ADS is defined with respect to unique distances. We can apply any symmetry breaking between nodes of equal distance, but to maintain efficiency, in particular for DP computation, we specify a particular one (this is all for analysis purposes). The symmetry breaking is defined according to the scan order of incoming edges to v in the representation of the graph. Amongst two nodes u and w so that $x = d_{uv} = d_{wv}$, we consider all paths of length x to v originating from u (or w) and associate with u the least ordered incoming edge to v. The closer one of u and w is the defined as the one with the earlier edge. If both have the same earliest edge (y, v), we consider the same order with respect to the common previous node y on the path, and so on. If DP performs relaxations according to this order, then we maintain the property that inserted nodes are in the final ADS (with respect to the "unique" order).

Parallelizing PRUNED DIJKSTRA'S.

As stated, the algorithm performs n sequential Dijkstra computations. The dependences can be improved. Consider $k = 1$: We partition the nodes to two sets according to rank. We then perform the computation from the set of lower rank nodes collapsed together to a single node. This will provide us ADS entries and their distances for the closest node in that batch. After we do this, we can proceed for the second batch without completely resolving the first set of nodes. Recursing, this gives us logarithmic depth. Further details are in [14].